Table of Contents

Chapter 1: Introduction to the Enstore Mass Storage System

1.1 About Enstore

Enstore is the mass storage system implemented at Fermilab as the primary data store for large data sets. Its design was inspired by the mass storage architecture at DESY, and it originates from discussions with the DESY designers. Enstore provides access to data on tape or other storage media both local to a user's machine and over networks. Enstore is designed to provide high fault tolerance and availability sufficient for the RunII data acquisition needs, as well as easy administration and monitoring. It uses a client-server architecture which provides a generic interface for users and allows for hardware and software components that can be replaced and/or expanded.

Enstore has four major kinds of software components:

- namespace (implemented by PNFS), which presents the storage media library contents as though the data files existed in a hierarchical UNIX file system
- dCache, a front-end data file caching system through which users typically access Enstore
- the Enstore servers, which are software modules that have specific functions, e.g., maintain database of data files, maintain database of storage volumes, maintain configuration, look for error conditions and sound alarms, communicate user requests down the chain to the tape robots, and so on
- **encp**, a program for copying files directly to and from the mass storage system

Enstore can be used from machines both on and off-site. Off-site users are restricted to accessing Enstore via dCache, and on-site users are encouraged to go through dCache as well.

Enstore supports both automated and manual storage media libraries. It allows for a larger number of storage volumes than slots. It also allows for simultaneous access to multiple volumes through automated media libraries. There is no preset upper limit to the size of a data file in the enstore system; the actual size is limited by the physical resources. The lower limit on the file size is zero. The upper limit on the number of files that can be stored on a single volume is about 5000.

Enstore allows users to search and list contents of media volumes as easily as they search native file systems. The stored files appear to the user as though they exist in a mounted UNIX directory. The mounted directory is actually a distributed virtual file system in PNFS namespace containing metadata for each stored file. Enstore eliminates the need to know volume names or other details about the actual file storage.

Users typically access Enstore via the dCache caching system. The protocols supported by dCache include dccp, gridftp (globus-url-copy), kerberized ftp and weakly-authenticated ftp (these are described in Chapter 4: *Using the dCache to Copy Files to/from Enstore*). On-site users may bypass dCache and use the **encp** program, the Enstore copy command roughly modeled on UNIX's **`cp`**, to copy files directly to and from storage media.

There are several installed Enstore systems at Fermilab. Currently these include CDFEN for CDF RunII, D0EN for D0 RunII, and STKEN for all other Fermilab users. Web-based monitoring for the Enstore systems is available at `http://hppc.fnal.gov/enstore/`. Currently, all storage libraries are tape libraries. The Computing Division operates and maintains the tape robots, slots, and other tape equipment, but for the present, experiments provide and manage their own volumes.

1.2 PNFS Namespace

PNFS is a virtual file system package that implements the Enstore namespace. It was written at DESY. PNFS is mounted like NFS, but it is a virtual file system only. It maintains file grouping and structure information via a set of tags in each directory. The **encp** program communicates this information between PNFS and the Enstore servers when it uploads or downloads a data file.

PNFS can only be mounted on machines that are physically at the lab. When a user copies a data file from disk to the Enstore system, he or she specifies its destination in terms of a PNFS directory. The data file gets copied to a storage volume (selected according to the tags of the specified PNFS directory) and a corresponding metadata entry is created in the PNFS directory. This entry takes the name given in the **`encp`** command line or in the protocol-specific dCache command. It contains metadata about the data file, including information about the file transfer, the data storage volume on which the data file resides, the file's location on the volume, and so on.

To browse file entries in the Enstore system, on-site users can mount their experiment's PNFS storage area on their own computers, and interact with it using standard non-I/O UNIX operating system utilities (see section 3.1 *UNIX*

Commands You can Use in PNFS Space). Normal UNIX permissions and administered export points are used for preventing unauthorized access to the name space.

1.3 Storage Groups

Each experiment or research project is assigned a unique *storage group* identifier by the Enstore administrators. Enstore uses the storage group names to control and balance assignment of resources, such as tape drives and media, among the experiments. Each storage group is assigned an area in PNFS, e.g., an experiment XYZ might be assigned the storage area `/pnfs/xyz`.

1.4 File Organization on Storage Media

1.4.1 File Family

Files are grouped on data storage volumes according to a *file family*[1] attribute. A file family is a name that defines a category, or family, of data files. Each experiment (i.e., each storage group) must carefully plan its set of file families. There may be many file families configured; by design there is no pre-set upper limit on the number. A given storage volume may only contain files belonging to one file family.

Every directory in `/pnfs` namespace has a file family associated with it. Every data file added to the Enstore system (i.e., every file for which an entry appears in the `/pnfs` namespace) is thus associated with the file family of the directory under `/pnfs` into which it was initially copied.

Associated with a file family are a *file family width* (an integer value), and a *file family wrapper* (a format specification). The file family, file family width, and file family wrapper for a PNFS directory are initially inherited from the parent directory. They may be reset as permissions allow, generally only by a small group of designated people in each experiment.

1. The grouping is really based on the triplet of quantities: storage group + file family + wrapper. However, most users have access to only one storage group, and use the default wrapper, so from the user's perspective, the only relevant attribute for file grouping is file family, typically.

1.4.2 File Family Width

File family width is an integer value associated with a file family that is used to limit write-accessiblity on data storage volumes. There is currently no width associated with reading. For a given media type and for a given file family, Enstore limits the number of volumes available for writing at any given time to the value of the file family width (except when unfilled volumes are already mounted for previous reads). Correspondingly, the number of media drives on which the volumes are loaded is also limited to the width.

1.4.3 File Family Wrapper

A file family wrapper specifies the format of files on the storage volume. It defines information that gets added before and after data files as they're written to media. In this way the data written to tape is self-contained and independent of metadata stored externally.

There are two wrapper types implemented, `cpio_odc` and `cern`. Currently (November 2003) all tapes are written using the `cpio_odc` wrapper. (For NULL volumes there is a null wrapper.)

- All files with the `cpio_odc` wrapper are dumpable via **`cpio`**. This wrapper has a file length limit of (8G – 1) bytes. It is sufficient for the vast majority of data files, as most files are still under 2GB.
- The `cern` wrapper accommodates data files up to (2^64 –1) bytes, which in effect limits the filesize to the tape length, since spanning and striping are not supported as explained in section 1.6 *Data Storage Volumes in Enstore*. It matches an extension to the ANSI standard, as proposed by CERN, and allows data files written at Fermilab to be readable by CERN, and vice-versa[1].

1.5 File Size Limitations

Enstore limits the size of a data file to the tape capacity, i.e., a single file cannot span more than one volume. The `cpio_odc` wrapper further limits the file size to (8G – 1) bytes, as mentioned in section 1.4.3 *File Family Wrapper*. Your OS may restrict your files to yet a smaller size.

☞ Note: PNFS can only represent a data file's size accurately up to (2G–1)B; beyond that, the file size is shown as 1. Enstore knows, stores and uses the real file size, so this PNFS display limitation does not pose a functionality problem.

1. Since CERN will allow files to span volumes, and Fermilab doesn't, users will not be able to use Enstore to read volumes from the CERN system that contain partial files.

A 1 byte file size indicated by PNFS indicates to the the user that the file is likely quite large. (You can use the **`enstore pnfs`** command with the **`--filesize`** option as described under section 8.4 *enstore pnfs* to find the actual file size.)

1.6 Data Storage Volumes in Enstore

Enstore is designed to support a variety of data storage media. Currently, only tapes have been implemented, and the information in this section has been written with tapes in mind. In principle, the information should apply equally to other media types; we will update this section as necessary when other media types are implemented.

1.6.1 Tape Features

Tapes are self-describing and exportable so that in the unlikely event of lost metadata in Enstore, a volume can be dumped, and the information retrieved.

Tapes are required to have ANSI Vol1 header labels. Labelling helps to easily identify each volume and/or test for a blank one, thereby inhibiting the inadvertent overwriting of used tapes. The Enstore admin needs to know whether tapes are labelled or completely blank when they are inserted into the library; the Enstore software can label tapes if necessary.

1.6.2 File Organization, Storage and Access

Data files are physically clustered on volumes according to each experiment's file family classification scheme. A given volume may only contain files belonging to one file family, one wrapper type, and one storage group.

A single file cannot span more than one volume[1]. When writing files to media, Enstore compares the size of the file it's ready to copy against the volume's remaining empty space in order to determine whether the file will fit. If the file is too large to fit, Enstore marks the volume as full, and writes the file to a different volume. Thus volumes are filled, modulo some fraction of a data file. (Volumes can be reopened if an administrator decides that too much space is left unused.)

Enstore supports random access of files on storage volumes, and also *streaming*, the sequential access of adjacent files.

1. Since files cannot span multiple volumes, striping is not supported either. Striping refers to files (usually large ones) being split onto two or more volumes, each writing simultaneously, in order to expedite the writing process.

1.6.3 Quantity of Volumes

Enstore allows data storage volumes to be “faulted out to shelf”, i.e., removed from a robot. This feature makes it possible for each experiment to have a larger number of volumes than it has slots in the robot, and in fact there is no limit on the number of volumes used by an experiment. To accommodate the unmatched numbers of volumes and slots, Enstore provides separate quotas in volumes and in slots.

Note that moving volumes in and out of the robot requires operator intervention, and should be minimized.

1.6.4 Import/Export of Volumes

Tapes can be generated outside the Enstore system and imported into it. Conversely, Enstore tapes can be dumped via standard UNIX utilities, thereby allowing them to be readable with simple tools outside the Enstore framework. The tools to do this are wrapper-dependent (e.g., tapes whose files have the cpio wrapper can be dumped via the **`cpio`** utility). Currently there is no utility (except **`dd`**) to dump a tape whose file family wrapper is cern.

1.7 dCache: The Enstore Front-End

1.7.1 Overview

The dCache has been designed as a front-end for a set of Hierarchical Storage Managers (HSMs), namely Enstore, EuroGate and DESY’s OSM. It can be viewed as an intermediate “relay station” between client applications and the HSM (Enstore, in our case). Client systems communicate with dCache via any of a number of protocols, listed in 1.7.3 *Protocols for Communicating with dCache*. DCache communicates with Enstore (in a manner transparent to the user) via a high-speed ethernet connection. The dCache decouples the potentially slow network transfer (to and from client machines) from the fast storage media I/O in order to keep Enstore from bogging down.

Data files uploaded to the dCache from a user’s machine are stored on highly reliable RAID disks pending transfer to Enstore. Files already written to storage media that get downloaded to the dCache from Enstore are stored on ordinary disks.

The dCache is installed at Fermilab on a server machine on which the `/pnfs` root area is mounted. Since PNFS namespace can only be mounted on machines in the fnal.gov domain,[1] off-site users may only access Enstore via the dCache. On-site users are strongly encouraged to go through the dCache as well. We discuss dCache in more depth in Chapter 4: *Using the dCache to Copy Files to/from Enstore*.

Read more general information about the dCache at the DESY site: `http://www-dcache.desy.de`.

1.7.2 Advantages

The principal advantages of using the dCache are:

- Optimized usage of existing tape drives due to transfer rate adaption.
- Possible usage of slower and cheaper drive technology without overall performance reduction.
- Optimized usage of the robot systems by coordinated read and write requests.
- Better usage of network bandwidth by exploring the best location for the data.
- No explicit staging required to access the data.
- Ability to do posix-like IO reads and writes to data files instead of transferring entire files.
- Working ROOT interfaces.
- Tapeless data methods, raw data to reconstruction to analysis to users.
- Written to tape as 'by-product'; no tape delays.
- Pnfs does not have to be mounted for access to the data.
- Same access to storage system, on and off site. Strong authentication, both gss and gsi to the data. Native and ftp access to the data.
- The access methods for data would be uniform, independent of data’s media location.
- Even without the back-end HSM (e.g., Enstore), the dCache system could be seen as a huge data store with a unique namespace and standardized access methods. Care will be taken that valuable data resides on safe disks as long as no HSM copy exists. Back-end storage to the HSM can be done regularly (policy based) or by manual intervention only.
- A joint DESY-FNAL effort makes the use of manpower more efficient and guarantees continued support and maintenance of the developed software.

1. There are some exceptions; arrangements for PNFS mounting have been made for some experiments whose systems are managed by the Computing Division, e.g., soudan.org for Minos.

1.7.3 Protocols for Communicating with dCache

Whenever an application needs to talk to the dCache, it has to choose an appropriate *door* into the system. There are a number of different dCache doors through which users/applications can send requests to Enstore. Doors are protocol converters from the dCache point of view, and they are responsible for strong authentication, as necessary. One door may be for Kerberized ftp read/write access, another for dcap (dCache native C API), gridftp, weakly authenticated ftp read-only access, and so on. Each experiment determines which door(s) its experimenters may use, and communicates this information to the Enstore administrators who manage the doors' configurations. Most doors are for native transfers, and are local. See Chapter 4: *Using the dCache to Copy Files to/from Enstore* for more information.

Chapter 2: Getting Ready to Use Enstore and DCache

2.1 Setting up Access to Enstore

2.1.1 Initial Steps for All Users

1) Find out what your volume quota is from your experiment's Enstore administrator, and make sure you reserve what you need, according to your experiment's procedures. The experiment's Enstore liaison should use the online form *STKEN Mass Storage Request Form* at `http://computing.fnal.gov/cd/forms/storagerequest.html` to request quota for the experiment.

2) Find out what area in `/pnfs` namespace your experiment uses.

3) Read about file families (see section 1.4.1 *File Family*), and find out from the people in your experiment responsible for implementing Enstore how file families have been configured for your experiment. Determine what file family(ies), and hence which subdirectories in `/pnfs` namespace, you want to write to and/or read from.

4) Encp and Enstore commands use whatever routing the client system or network administrator sets between the client system and the Enstore system for data transfers. If you (as the sysadmin or network admin of the large client machine) want to restrict the set of interfaces that encp/Enstore uses, you need to create the file `enstore.conf`. This file controls the interface-router mapping for the network connections used by encp/Enstore. For information and instructions, see Appendix A: *Network Control*.

5) Navigate to the Enstore monitoring system web page, titled *Fermilab Mass Storage System*, at `http://hppc.fnal.gov/enstore/`. Select the Enstore system that your experiment uses, and browse the system information for it. You might want to bookmark this page.

6) Subscribe to the *stk-users@fnal.gov* listserv mailing list for announcments about Enstore and the STKEN Enstore system. D0 users, subscribe to *d0en-announce@fnal.gov* and *sam-admin@fnal.gov*. CDF

users, subscribe to *cdfdh_oper@fnal.gov.*

☞ Notes:

- Data are moved with the default TCP window size on the machine. There is a potential for an extreme performance degradation if the default window is set too large. A value of about 32K works well at most locations at Fermilab.
- It is recommended to keep the number of files in any given directory under 2000.

2.1.2 Further Steps for non-dCache Users Only

1) Make sure your node and network can provide adequate throughput. To determine the optimal data transfer rate, consult the Enstore administrators.

2) See if your experiment's `/pnfs` area is mounted on your machine, by using standard UNIX utilities like **`cd`** and **`ls`**. If it's already mounted, skip to step (6). If not, continue.

3) Check to see if authorization has been granted to mount the `/pnfs` area on the machine you plan to use. To do so:

   a) Go to the *PNFS Exports Page*[1], at `http://www-<xyz>en.fnal.gov:/enstore/pnfsExports.html`, where `<xyz>` is one of `stk`, `d0` or `cdf`, depending on the Enstore system used by your experiment.

   b) Scroll down to the *PNFS ExportList Fetch Begin: <date/time>* area, and look for your node and `/pnfs/storage-group` area. If they're listed, authorization has been granted; skip to (5). If not, continue.

4) Notify your experiment's Enstore liaison that you need authorization to mount the `/pnfs` area on the machine you plan to use. He or she will need to send your request on to *enstore-admin@fnal.gov.*

5) Once authorization has been granted, mount the `/pnfs` area on your machine if you have root permission, or send a request to the machine's system administrator to mount it. To mount the area yourself, edit the `/etc/fstab` file and add a line with the following strings (they should appear all on the same line in the file; we separate them into six

1. Note that you can also get to the *PNFS Exports Page* from the *Fermilab Mass Storage System* web page at `http://hppc.fnal.gov/enstore/` via the following path: Under *Installed Enstore Systems*, click the system your experiment is using. On the *Enstore System Status* page, click "Log Files". Under *User Specified Log Files*, click "PNFS Export List" to arrive at the *PNFS Exports Page*.

lines here for clarity):

```
remote_enstore_server_node:enstore_server_directory
/pnfs/local_mount_point
mount type
comma_separated_attributes
0
0
```

where the 0 in the 2nd-to-last line means no dump of filesystem, and the 0 in the last line means no fsck checks at boot time. For example:

```
stkensrv1:/E872 /pnfs/E872 nfs user,intr,bg,hard,rw,noac 0 0
```

Usually, `local_mount_point` is the same as `enstore_server_directory`. Make sure that `local_mount_point` exists! (A typical error message is "backgrounding".)

6) Install UPS/UPD on your system. See Part III of the UPS/UPD manual at `http://www.fnal.gov/docs/products/ups/ReferenceManual/parts.html#partIII`.

7) Install the **encp** product on your machine (see below).

Installing Encp

To install the **encp** product from KITS using UPD, run:

```
$ setup upd
```

```
$ upd install -G "-c -q <xyz>" encp
```

where **`<xyz>`** stands for one of the Enstore systems. Currently, these include:

`stken`	for general Fermilab users
`d0en`	for D0 users
`cdfen`	for CDF users

For example, a CDF experimenter would type:

```
$ upd install -G "-c -q cdfen" encp
```

2.2 Important Environment Variables

There are two important environment variables that are generally set in the Enstore setup script. Users who work on more than one Enstore system (e.g., stken and cdfen) at a time in different windows may need to know about these in case they use the wrong window for a particular Enstore system!

The variables are:

$ENSTORE_CONFIG_HOST

points to the Enstore server that is running the configuration server (see section 7.6 *Configuration Server*).

☞ All production systems currently use srv2 (i.e., cdfensrv2, d0ensrv2 or stkensrv2) as the $ENSTORE_CONFIG_HOST computer. This is different from the computer from which the pnfs filesystem is mounted (which is srv1, i.e., cdfensrv1, d0ensrv1, stkensrv1).

$ENSTORE_CONFIG_PORT

sets the port number; the value is (by convention) 7500 for all installations at Fermilab.

Chapter 3: PNFS Namespace

3.1 UNIX Commands You can Use in PNFS Space

Data files do not actually reside in `/pnfs` namespace, and errors occur on attempts to read or write the content of the files, or to manipulate the content. Therefore, UNIX commands such as **`cat`**, **`more`**, **`less`**, **`grep`**, **`head`**, **`tail`**, **`wc`**, **`od`**, **`file`**, **`cp`**, and so on, fail if you run them on files listed under `/pnfs`. However, virtually any non-I/O UNIX command can be used in `/pnfs` namespace. For these commands, the standard options work in the standard way. Commands that you may find useful include:

• ls	• pwd
• mv and mvdir	• find
• rm and rmdir	• cd
• mkdir	• ln (hard links only)[a]
• stat[b]	

a. For ln, hard links must be used to ensure that all the metadata information is linked; symbolic links do not work properly .
b. Stat is not available in all operating systems.

3.2 About PNFS Tags

Before files can be written to tape, Enstore needs to know where and how to write them. Pnfs uses tag files (usually just called tags) in the `/pnfs` namespace to specify this type of configuration information, and **encp** transfers this information to Enstore. Tags are associated with directories in the `/pnfs` namespace, not with any specific file, and thus apply to all files within a given directory. As a new directory in the `/pnfs` namespace is created, it inherits the tags of its parent directory. Allowable characters within tags are: alphanumeric characters, underscore (`_`), dash (`-`), and slash (`/`).

3.2.1 Tag Listing

The tags include:

file_family	This tag determines the file family associated with all files in this directory. See section 1.4.1 *File Family* for information on file families.
file_family_width	This tag determines the file family width associated with all files in this directory. See section 1.4.2 *File Family Width* for information on file family width.
file_family_wrapper	This tag determines the file family wrapper associated with all files in this directory. See section 1.4.3 *File Family Wrapper* for information on file family wrappers. The default is `cpio_odc`.
library	This tag determines the virtual library (and thus the library manager) associated with all files in this directory. See section 7.3 *Library Manager* for information about the library.
storage_group	This tag determines the storage group associated with all files in this directory, and shows up as your experiment's top level directory under `/pnfs`. Typically, one storage group is associated with an entire experiment. A storage group is assigned to each experiment by the Enstore administrators. Users never change this tag.

3.2.2 How to View Tags

Off-site users cannot mount pnfs, and therefore cannot see tags. On-site users: to see the values of the tags for a given directory, first setup **encp** (with qualifier, see section 5.1 *Setup encp*) then **`cd`** to the `/pnfs` subdirectory of interest (or enter the directory as an argument to `--tags`) and enter the command:

```
% enstore pnfs --tags

    .(tag)(file_family) = dcache
    .(tag)(file_family_width) = 1
    .(tag)(file_family_wrapper) = cpio_odc
    .(tag)(library) = eagle
    .(tag)(storage_group) = test
    -rw-rw-r--   11 xyz     sys           6 Jul 26 10:22 .(tag)(file_family)
    -rw-rw-r--   11 xyz     sys            1 May  5  2000 .(tag)(file_family_width)
    -rw-rw-r--   11 xyz     sys            8 May  5  2000 .(tag)(file_family_wrapper)
    -rw-rw-r--   11 xyz     sys            5 May  5  2000 .(tag)(library)
    -rw-r--r--   11 xyz     sys            4 Jul 26 10:20 .(tag)(storage_group)
```

The output first lists the tags and their values, then the tags again in long format to show the owners and protection modes.

Chapter 4: Using the dCache to Copy Files to/from Enstore

Whenever a client application needs to talk to the dCache, it has to choose an appropriate *door* into the system. For each door, there are corresponding utilities for copying files back and forth between your machine and your `/pnfs/storage-group` area on the machine running dCache. We describe how to use the supported utilities in this chapter.

Currently (November 2003), there are four Fermilab dCache server nodes, three corresponding to Enstore installations (FNDCA, CDFDCA, and D0DCA), and CMSDCA for CMS. Each dCache server may have multiple doors, thus allowing a variety of access methods. Each door is limited to about 50 simultaneous transfers; more doors can be added as needed. The dCache supports Kerberos V5 for FTP, the dCache native dCap C-API, and GSI FTP.

☞ The dCache server node and the ports documented in this section are subject to change. You can always find the current configuration from the web page `http://www-isd.fnal.gov/enstore/dcache_user_guide.html`.[1]

4.1 DCache-Native dCap

4.1.1 About dCap

DCap is a dCache-native access protocol. It is available in KITS at `ftp://fnkits.fnal.gov/products/dcap/`. The libdcap library provides POSIX-like **open, create, read, write** and **lseek** functions to the dCache storage. In addition there are some specific functions for setting debug level, getting error messages, and binding the library to a network interface. See `http://www-dcache.desy.de/manuals/libdcap.html` for usage information.

1. It is available from the *Fermilab Mass Storage Systems* home page (`http://hppc.fnal.gov/enstore/`); see the list of items under *Documentation* for dCache, and use the *User Access at FNAL* link.

If your dCap door uses Kerberos V5 authentication, first obtain a Kerberos principal for the FNAL.GOV realm, if you don't already have one. Install the dCap product on your computer. See `http://www-dcache.desy.de/manuals/dcap_setup.html`.

The nodes and ports available for dCap are subject to change; to get a current listing, run the following command, using your storage group (sample output shown for storage group cdfen):

```
% cat '/pnfs/cdfen/.(config)(dCache)(dcache.conf)'

   cdfdca.fnal.gov:25125
   cdfdca.fnal.gov:25136
   ...
   cdfdca2.fnal.gov:25153
   cdfdca2.fnal.gov:25154
   cdfdca3.fnal.gov:25155
   ...
```

The dCap protocol requires specification of the dCache server host, port number, and domain, in addition to the inclusion of "**`/usr`**" ahead of the storage group designation in the PNFS path. Its structure is shown here:

```
dcap://<serverHost>:<port>/</pnfs>/<storage_group>/usr/<filePath>
```

There are supposed to be two slashes inbetween the port number and `pnfs`, e.g., `... :24124//pnfs/...`, but since users frequently just put one slash, we've allowed either one or two.

4.1.2 The dccp Command

The command **`dccp`**, which provides a **`cp`**-like functionality on the PNFS file system, is available in the dCap product. The **`dccp`** command has the following syntax:

or, more simply:

```
% dccp [ options ] source_file [ destination_file ]
```

The options and command usage are described at `http://www-dcache.desy.de/manuals/dccp.html`.

A useful related command is:

`dc_stage [-t <number of seconds>] source [ dest]`

This prestages the request; for read requests only. It is particularly useful when you'd like to grab the file quickly from the dCache when you're ready for it. Use this with the **`-t`** option to set an interval of time between the download to the dCache and the download from the dCache to your local system. If **`-t`** is not used, the default interval is zero.

☞ If you run a **dccp** command and it fails because the port is unavailable, try the command again with a different port number, or with a different host and port combination.

Syntax and Examples (PNFS Not Mounted Locally)

If PNFS is not mounted locally (the general case), you'll have to supply the protocol, node, port, and pnfs directory for the remote location (the "source" on reads, and the "destination" on writes). For example, a command requesting a write to Enstore would have this structure:

```
% dccp path/to/local/file \
dcap://<serverHost>:<port>/</pnfs>/<storage_group>/usr/<filePath>
```

Here is an example of this, requesting a write from your local `/tmp` directory:

```
% dccp /tmp/myfile \
dcap://cdfdca.fnal.gov:25140//pnfs/fnal.gov/usr/cdfen/x/myfile
```

To check if a file is on disk in the dCache, run **`dc_check`**:

```
% dc_check \
dcap://fndca.fnal.gov/:24124/pnfs/fnal.gov/cdf/myfile
```

If this were to be a read rather than a write, it would look like:

```
% dccp \
dcap://cdfdca.fnal.gov:25140//pnfs/fnal.gov/usr/cdfen/x/myfile\
/tmp/myfile
```

To pre-stage this same request with an hour interval, use **`dc_stage`**:

```
% dc_stage -t 3600 \
dcap://cdfdca.fnal.gov:25140//pnfs/fnal.gov/usr/cdfen/x/myfile\
/tmp/myfile
```

Syntax and Examples (PNFS Mounted Locally)

If PNFS is mounted on your local machine, you only need to specify the simple PNFS path of the remote file, e.g. (for a write):

```
% dccp path/to/local/file /pnfs/<filePath>
```

For example (using the same file as in the previous examples):

```
% dccp /tmp/myfile /pnfs/cdfen/x/myfile
```

will write the file to Enstore, and the following will read it from Enstore and put it into your local `/tmp` directory:

```
% dccp /pnfs/cdfen/x/myfile /tmp/myfile
```

4.2 Grid (GSI) FTP

GSI stands for Grid Security Interface. GSI FTP uses Grid Proxies for authentication and authorization and is compatible with popular Grid middleware tools such as globus-url-copy (from the Globus toolkit available at `http://www.globus.org` or from sam_gridftp in Kits). The dCache GSI FTP currently runs on port 2811 on the following nodes (different nodes for different user groups):

- fndca.fnal.gov, port 2811 (for general users)
- cdfdca, port 2811 (for CDF)
- d0dca, port 2811 (for D0)
- cmsdca, port 2811 (for CMS)

It is more convenient to run this through an interface like srmcp (see section 4.2.4 *Storage Resource Management (SRM)*) which allows you to perform multiple transfers in a single command. In addition, it optimizes the parameters of the transfer, and allows FTP to scale with user load (overcoming a passive gridftp protocol issue).

4.2.1 Obtain Grid Proxies

Globus tools require that a user be authenticated with a short-term authentication Grid proxy. This proxy can be created from (long-term) X.509 credentials issued by DOE science grid (or other Certificate Authority listed on `http://computing.fnal.gov/security/pki`) or from Kerberos credentials at Fermilab. A proxy expires after a preset duration, and then a new one must be regenerated from the user's (long-term) X.509 certificate.

X.509 Grid proxies can be issued automatically for Fermilab users authenticated to Kerberos. See `http://computing.fnal.gov/security/pki/` for instructions. This involves downloading a KX.509 certificate. KX.509 can be used in place of permanent, long-term certificates. It works by creating X.509 credentials (certificate and private key) using your existing Kerberos ticket. These credentials are then used to generate the Globus proxy certificate. KX.509 is described at `http://www.ncsa.uiuc.edu/~aloftus/NMI/kx509.html`.

For non-Fermilab people, Grid proxies typically must be created from X.509 certificates. See `http://www.doegrids.org/pages/cert-request.htm`.

4.2.2 GSI FTP with globus-url-copy

Install the Globus toolkit (available from a variety of locations, `http://www.globus.org` is one). Then run the **`globus-url-copy`** command in order to use the GSI FTP protocol to transfer files. Use the **`gsiftp://`** URL prefix for the PNFS (Enstore) path, and **`file://`** for the other URL.

E.g., to copy **from** Enstore the syntax is:

```
% globus-url-copy
  gsiftp://[[<src_node>:]port]/<source_url_path>
  file://[[<dest_node>]:port]/<dest_url_path>
```

and to copy **to** Enstore, it's:

```
% globus-url-copy file://[[<src_node>:]port]/<source_url_path>
  gsiftp://[[<dest_node>]:port]/<dest_url_path>
```

In the case of a CDF user copying from Enstore to a local disk, this would look like:

```
% globus-url-copy gsiftp://cdfdca.fnal.gov:2811/<pnfs_path>
  file:///<local_url_path>
```

A D0 user copying from a remote disk to Enstore would use a command like this:

```
% globus-url-copy file://<remotenode>:<port>/<remote_url_path>
  gsiftp://d0dca.fnal.gov:2811/<pnfs_path>
```

You can also copy from one Enstore system to another, e.g., from CDFDCA to FNDCA.

```
% globus-url-copy gsiftp://cdfdca.fnal.gov:2811/<pnfs_path>
  gsiftp://fndca.fnal.gov:2811/<pnfs_path>
```

4.2.3 GSI FTP with Kftpcp

GSI FTP is also available with kftpcp (see section 4.4 *Kerberized FTP via the kftpcp Command*). Install and setup kftp (from Kits `ftp://fnkits.fnal.gov/products/`). Also from kits, install and setup gsspy_gsi (for Grid proxy) instead of gsspy_krb. Kftpcp works the same as described in section 4.4 except that the port number is 2811 in this case.

We refer you to section 4.4 for details, but here's a quick example for a general user (using STKEN) to copy from Enstore to a local disk:

```
% kftpcp -p 2811 -m p [-v] \
  [<your_login_id>@]fndca:<pnfs_path> \ </path/to/local_file>
```

4.2.4 Storage Resource Management (SRM)

SRM is middleware for managing storage resources on a grid. The SRM implementation within the dCache manages the dCache/Enstore system. It provides functions for file staging and pinning[1], transfer protocol negotiation and transfer url resolution.

The SRM client **srmcp** provides a convenient way to transfer multiple files from/to Enstore via dCache using a variety of protocols.

To read about SRM, go to `http://sdm.lbl.gov/`, click on **PROJECTS**, and look for *Storage Resource Management (SRM) Middleware Project*.

Srmcp is the implementation of SRM client as specified by the SRM spec (see `http://sdm.lbl.gov/srm/documents/joint.docs/srm.v1.0.doc`). You can use **srmcp** for the retrieval and/or storage of files to/from Enstore (or other Mass Storage Systems which implement SRM, e.g., SLAC's, CERN's). In this document we focus on file transfers to/from Fermilab's Enstore via dCache.

Preparing to Use srmcp

Two packages are available, one with java (srmcp), the other with a C-based client (srmtools); they are both in Kits (`ftp://fnkits.fnal.gov/products/`). To use the java-based **srmcp**, you will need to install java on your system. You will also need to install either the globus toolkit or dccp, depending on which protocol you wish to use. In order to use GSI with **srmcp**, follow the instructions in the README.SECURITY file that comes with srmcp v1_2 in Kits.

Command Syntax

```
% srmcp [options] source(s) destination
```

Default options will be read from a configuration file but can be overridden by command line options. The options are listed and defined in the srmcp v1_2 README file in Kits. We do not list them here.

The SRM protocol, used for the remote file specification, requires the SRM server host, port number, and domain. For the fnal.gov domain, the inclusion of "**`/usr`**" ahead of the storage group designation in the PNFS path is also required. Its structure is shown here:

```
srm://<serverHost>:<portNumber>/<root of fileSystem>
  /<storage_group>[/usr]/<filePath>
```

Some examples, the first two for the fnal.gov domain, the third for cern.ch:

1. Pinning refers to making a file undeletable in the cache for the period of time called the "lifetime of the job".

- **srm://cdfdca.fnal.gov:25129//pnfs/fnal.gov/usr/cdfen/filesets/<filePath>**
- **srm://fnisd1.fnal.gov:24129//pnfs/fnal.gov/usr/cdfen/<filePath>**
- **srm://wacdr002d.cern.ch:9000/castor/cern.ch/user/<filePath>**

Examples

These examples are taken from the srmcp v1_2 README file in Kits (with unnecessary options removed).

The following command will retrieve two files `/mypath/myfile1.ext` and `/mypath/myfile2.ext` from Enstore via dCache (for a CDF user) and store them in the user's local directory `/home/me/targetdir`:. Notice that srmcp requires that the PNFS path include `/pnfs/fnal.gov/usr/` ahead of the storage group designation.

```
% srmcp \
  srm://cdfdca.fnal.gov:25129//pnfs/fnal.gov/usr/cdf/myfile1.ext \
  srm://cdfdca.fnal.gov:25129//pnfs/fnal.gov/usr/cdf/myfile2.ext \
  file://localhost//home/me/targetdir
```

The following will copy the same files from one Enstore installation (CDFEN) to another (STKEN):

```
% srmcp \
  srm://cdfdca.fnal.gov:25129//pnfs/fnal.gov/usr/cdf/myfile1.ext \
  srm://cdfdca.fnal.gov:25129//pnfs/fnal.gov/usr/cdf/myfile2.ext \
  srm:/fndca.fnal.gov:24128/targetdir
```

The following will get the file using dccp client, overriding the default (dccp would have to be already installed on you machine)[1]:

```
% srmcp  \
  -protocols=dcap   \
  srm:/fndca.fnal.gov:24128//pnfs/fnal.gov/usr/targetdir/myfile1.ext
  file:////tmp/myfile1.ext\
```

4.3 Simple Kerberized FTP

The dCache door for Kerberized ftp service enforces Kerberos authentication (see *Strong Authentication at Fermilab Documentation* at `http://computing.fnal.gov/docs/strongauth/`). It currently runs on the following nodes and corresponding ports:

- fndca.fnal.gov, port 24127 (for STKEN)

1. The four slashes in the last line refer to: `file://`; host, which comes next, is " "; path is `/tmp/...`.

- cdfdca, port 25127 (for CDFEN)
- d0endca, port 24127 (for D0EN)

(The port number is installation-specific.) Any Kerberized ftp client can be used on the client machine. You must specify the host port in your ftp command.

Notes:

- File read and write functionality is supported when the user (a) is authorized by the experiment to access the data stores, and (b) has obtained Kerberos credentials.
- Portal Mode (CRYPTOCard) access is not supported since it is not compatible with automated transfers or future GRID development.

4.3.1 Prepare to use Kerberized FTP

In order to establish the kftp service on dCache, you must first:

- have a valid Fermilab UNIX account (UID and GID)
- have a Kerberos principal for FNAL.GOV (if Kerberized access is required)
- ask your experiment's Enstore liaison to register you for the service; you'll need to provide the following information to the liaison:
  - · username
  - · UID and GID (run the command **`id`** at the UNIX prompt to find their values)
  - · storage group
  - · root path under `/pnfs/<storage_group>/...`
  - · if applying for Kerberized door, provide Kerberos principal(s)
  - · if applying for weak door, request a password by emailing dcache-admin@fnal.gov. This is for groups, not individuals.
- install the **kftp** product from KITS (optional; useful for running scripts to transfer files). To do so, run:

```
$ setup upd
$ upd install -G "-c" kftp
```

4.3.2 Sample Kerberized FTP session

User is authenticated to Kerberos and authorized for the Kerberized dCache door (currently at fndca.fnal.gov, port 24127):

```
% ftp fndca.fnal.gov 24127

    Connected to stkendca3a.fnal.gov.
```

```
220 FTPDoorIM+GSS ready
334 ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded
Name (fndca:aheavey):
200 User aheavey logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd aheavey/test3
250 CWD command succcessful. New CWD is </aheavey/test3>
ftp> ls
200 PORT command successful
150 Opening ASCII data connection for file list
dupl2
duplexps
226 ASCII transfer complete
ftp> get duplexps
local: duplexps remote: duplexps
200 PORT command successful
150 Opening BINARY data connection for /pnfs/fs/usr/test/aheavey/test3/duplexps
226 Closing data connection, transfer successful
42 bytes received in 0.033 seconds (1.2 Kbytes/s)
ftp>
```

4.4 Kerberized FTP via the kftpcp Command

In order to access data from a batch job or a background process, you should either use ftp client libraries (available from many sources), or the **kftp** package. This package includes a Kerberized client library and a GSI client library; you can use either. A regular ftp client (Kerberized or not) is an interactive program which is hard to use in batch mode.

See section 4.3.1 *Prepare to use Kerberized FTP* for installation information. To use the product in a UPS environment as a Kerberized FTP client, first run:

```
% setup gsspy_krb; setup kftp
```

Then run the **kftpcp** command to copy one or more files. This command can be used from the shell or in a script.

4.4.1 Syntax and Options

```
% kftpcp [<options>] <source_file> <destination_file>
```

The available options include:

-p <port>	ftp server port number
-m <a|p>	ftp server mode; active (default), or passive
-v	verbose mode

Notes:

- If your login id is the same on fndca and your local system, and if they match your Kerberos principal, you can leave off **`<your_fndca_login_id>@`** in front of **`fndca:`** in the command.
- Depending on how your access is configured, typically you only need to specify the path to the remote file starting from the directory under your `/pnfs/<storage_group>/` area. E.g., to specify the remote file `/pnfs/my_storage_group/path/to/file` on the command line, enter only `/path/to/file`, including the initial slash. You can use the full specification (starting with `/pnfs/<domain>/usr/<storageGroup>`)

4.4.2 Download a File

To download a stored data file from Enstore via the dCache, using fndca as a sample server host, run:

```
% kftpcp -p 24127 -m p [-v] \
  [<your_fndca_login_id>@]fndca:</path/to/remote_file> \
  </path/to/local_file>
```

4.4.3 Upload a File

To upload a new data file, again using fndca, run:

```
% kftpcp -p 24127 -m p [-v] \
  </path/to/local_file> \
  [<your_fndca_login_id>@]fndca:</path/to/remote_file>
```

4.4.4 Examples

To read (download) the stored file `/pnfs/storage_group/mydir/myfile` into a local file of the same name, run:

```
% setup kftp
```

```
% kftpcp -p 24127 -m p -v myloginid@fndca:/mydir/myfile \
  /path/to/myfile
```

```
   Transferred 42 bytes
```

Or, if your usernames and principal all match, you could shorten it to:

```
% kftpcp -p 24127 -m p -v fndca:/mydir/myfile /path/to/myfile
```

4.5 Weakly-Authenticated FTP Service (Read-only)

The dCache weakly-authenticated ftp service currently runs on node the following nodes and corresponding ports:

- fndca.fnal.gov, port 24126 (for STKEN).
- cdfdca, port 25126 (for CDFEN)
- d0endca, port 24126 (for D0EN)

This is read-only, and is not necessarily allowed by all experiments. This ftp service can be accessed by ordinary ftp client software. You must specify the host port in your ftp command, as shown below. The Enstore admin will have sent you an email to confirm your registration for this service, and included a password for it.[1] This is a weak password. Log in with your username and password.

Sample weakly-authenticated read-only ftp session

Here we explicitly use a weakly-authenticated ftp client, `/usr/bin/ftp`, and make the connection to fndca port 24126. In the session, we first successfully retrieve a file called `myfile`, and secondly attempt to write a file `trace.txt` and (correctly) fail.

`% /usr/bin/ftp fndca.fnal.gov 24126`

```
Connected to stkendca3a.fnal.gov.
220 FTPDoorIM+PWD ready (read-only server)
Name (fndca:aheavey):
331 Password required for aheavey.
Password: (password entered here)
230 User aheavey logged in
ftp> cd aheavey/test3
250 CWD command succcessful. New CWD is </aheavey/test3>
ftp> ls
200 PORT command successful
150 Opening ASCII data connection for file list
myfile
myfile2
myfile3
226 ASCII transfer complete
10 bytes received in 0.018 seconds (0.55 Kbytes/s)
ftp> get myfile
200 PORT command successful
150 Opening BINARY data connection for
  /pnfs/fs/usr/test/aheavey/test3/myfile
226 Closing data connection, transfer successful
local: myfile remote: myfile
42 bytes received in 0.05 seconds (0.82 Kbytes/s)
ftp> put trace.txt
200 PORT command successful
500 Command disabled
```

1. If you need to change this password, send email to dcache-admin@fnal.gov.

```
ftp> bye
```

Chapter 5: Copying Files with Encp

Encp is an end-user command used to copy data files from disk to storage media and vice-versa. Its use is being discouraged in favor of the dCache, however we document it here for completeness.

Encp is maintained in KITS and in AFS product space as a separate product from Enstore, and is designed to be used in conjunction with it. Encp does not support recursive copies of data to and from Enstore; ensync is provided as a wrapper to encp for that purpose when writing to Enstore, see section Chapter 6: *Copying Directory Structures with Ensync*. Encp can copy multiple files to a single directory only. Encp can be used only from on-site machines in the fnal.gov domain. For off-site use, see section Chapter 4: *Using the dCache to Copy Files to/from Enstore*.

In this chapter, we assume you have UPS/UPD running on your local machine.

5.1 Setup encp

To setup **encp**, run the command:

```
% setup -q <qualifier> encp
```

where **`<qualifier>`** stands for one of the Enstore system hosts. Currently, these include:

`stken`	for general Fermilab (and CMS) users
`d0en`	for D0 users
`cdfen`	for CDF users

For example, a CDF experimenter would type:

```
$ setup -q cdfen encp
```

If you don’t specify the qualifier, the environment variable ENSTORE_CONFIG_HOST may get set to the wrong value. Check that ENSTORE_CONFIG_HOST specifies the correct server.

5.2 Encp Command Syntax and Usage

Encp plays the same role in the Enstore system that **`cp`** plays in UNIX. The syntax is:

```
% encp [<options>] <source_file> <destination_file>
```

With the exception of the option **`--help`**, we defer the list and definitions of options to section 5.5 *Encp Command Options*, and instead proceed with usage information.

Use the **`--help`** option to request the option listing for **encp** (we give the option listing in section 5.5 *Encp Command Options*), or the **`--usage`** option for syntax information:

```
% encp --usage
```

```
encp [OPTIONS]... <source file> <destination file>
encp [OPTIONS]... <source file> [source file [...]]\
 <destination>
```

5.3 Copy Files to and from Enstore Media

5.3.1 Run encp

First, setup **encp** (using the **`-q`** flag). You can use filename expansion (wildcard characters to specify a group of files). We recommend, however, that you copy one file at a time. Run the command as follows to copy a file to Enstore:

```
% encp [<options>] /<path-to>/.../<localfilename> \
 /pnfs/<storage-group>/.../<targetdir>/<remotefilename>
```

The presence of **`/pnfs/`** in the destination path indicates that this is a copy to the Enstore system (see section 1.2 *PNFS Namespace*). To copy from Enstore, change the source and destination file specifications, e.g.,:

```
% encp [<options>] \
 /pnfs/<storage-group>/.../<targetdir>/<remotefilename> \
 /<path-to>/.../<localfilename>
```

5.3.2 Examples

1) Standard copy to Enstore; no options. Copy `myfile` to the directory

`/pnfs/expt1/subdir/`:

```
% encp /path/to/myfile /pnfs/expt1/subdir/
```

2) Standard copy; no options. Download `/pnfs/expt1/subdir/myfile` to a different local directory from the cwd, and change the filename:

```
% encp /pnfs/expt1/subdir/myfile \
/other/local/dir/newfilename
```

3) Request the process to output some information to screen (**--verbose**). Again, copy `myfile` to the directory `/pnfs/expt1/subdir/`:

```
% encp --verbose 3 /path/to/myfile \
/pnfs/expt1/subdir/
```

4) Copy all the files in the cwd starting with the string `trigger1` to the directory `/pnfs/expt1/subdir/`:

```
% encp ./trigger1* /pnfs/expt1/subdir/
```

5) Copy all the files in `/pnfs/expt1/subdir/` starting with the string `trigger1` to the cwd:

```
% encp /pnfs/expt1/subdir/trigger1* .
```

5.4 More about Encp

5.4.1 Preventing Unwanted Overwriting

When an encp job starts, it first creates a zero length output file for every input file. In this way it reserves the necessary filenames and thus prevents another party from starting a competing encp process which would clobber the first.

5.4.2 Killing an encp Job

There are four traditional ways to abort a process:

• Ctrl-C (SIGINT)

• Ctrl-\ (SIGQUIT)

• kill (SIGTERM)

• kill -9 (SIGKILL)

The first three result in encp removing any remaining zero length files (as discussed directly above). With a “kill -9”, no cleanup occurs. For multi-file transfers, files successfully transferred before the signal is caught will be left alone.

5.4.3 Isolating Source of Bottlenecks

Encp (as of v3_1) supports isolating the rate transfers in the tape, disk and network via the option **`--threaded`** used in conjunction with the option **`--verbose`** with a value of 1 or higher. If **`--threaded`** is not specified, then the network and disk rates are calculated the same way as before, and display the same value as one another.

Here is an example without **`--threaded`** (with off-topic output removed for brevity):

```
% encp --verbose 1 /pnfs/xyz/10MB_002 /tmp/myfile
```

with output:

```
...
Transfer /pnfs/xyz/10MB_002 -> /tmp/myfile: 10485760 bytes
copied from ‘TEST01’ at 1.57  MB/S
(1.67 MB/S network) (2.87 MB/S drive) (1.67 MB/S disk)
...
Completed  transferring  10485760  bytes  in  1  files  in
14.2875500917 sec.
Overall rate = 0.7 MB/sec. Drive rate = 2.87 MB/sec.
Network rate = 1.67 MB/sec. Exit status = 0.
```

Note in the above output, the network and disk rates are the same.

Here is an example with **`--threaded`** and **`verbose 1`** (abbreviated output); note that the rates are separated, so that you can see where the bottleneck is (the disk, in this case):

```
% encp --verbose 1 --threaded /pnfs/xyz/10MB_002 /tmp/myfile
```

It produces output:

```
...
Transfer /pnfs/xyz/10MB_002 -> /tmp/myfile:
10485760 bytes copied from ‘TEST01’ at 2.41  MB/S
(8.09 MB/S network) (9.36 MB/S drive) (2.71 MB/S disk)
...
Completed  transferring  10485760  bytes  in  1  files  in
14.9129179716 sec.
Overall rate = 0.671 MB/sec. Drive rate = 9.36 MB/sec.
Network rate = 8.09 MB/sec. Exit status = 0.
```

The network and drive each have rates above 8 MB/s, and the disk rate is only 2.71 MB/s.

5.4.4 Encp Error Handling

Encp has functionality to retry and resubmit requests, where we distinguish between these two terms. Encp will *retry* (i.e., resend) a request after an error occurs. Encp will *resubmit* a request if it has been waiting for a mover for over 15 minutes; this is not due to an error condition but rather to keep queues current regardless of the server condition.

There are two general classifications of errors in encp: those that can be retried and those that can't. Three "retriable" errors can occur before the error "TOO_MANY_RETRIES" occurs.

The most common nonretriable errors include:

NOACCESS	the system has marked the volume as "potentially" bad
NOTALLOWED	an enstore administrator has marked a tape as unavailable for user access
USERERROR	usually is a file accessibility problem (doesn't exist, has wrong permissions, etc.)

Among the less common ones, there are:

VERSION_MISMATCH	the encp version is no longer compatible with the running Enstore system
CRC_MISMATCH	indicates a corruption error

Ask your Enstore administrator if you see others.

5.4.5 Finding files in different Enstore systems

File reads:	When reading from Enstore, encp can determine whether the current value of $ENSTORE_CONFIG_HOST (see section 2.2 *Important Environment Variables*) is pointing to the Enstore system that contains the requested file. If it points to the wrong one, encp will try the other Enstore installations to find the requested file. If the file is found, encp will retrieve the file; if the file is not found on any Enstore system, an error is returned to the user.
File writes:	When writing to Enstore; the value of $ENSTORE_CONFIG_HOST is always used.

5.4.6 Order of Processing Queued Requests

For reads, files are sorted out by volume. When all files from a single volume are complete, the next volume's files are requested.

For writes, one file at a time is submitted to the library manager. The order is that in which the files are specified on the command line. The tape is kept mounted during file writes on a best-effort basis. See Chapter 10: *Job Priority and Queue Management* for more information.

5.4.7 Calculating the CRC of a Local File

There is a small program called **ecrc** that calculates the CRC of a local file. It is used in this way:

```
% ecrc <filename>
```

For example,

```
$ ecrc ~/test_files/10MB_002
```

```
   size 10485760 buf_size 1048576 blocks 10 rest 0
    CRC 1294565006
```

To see what CRC information Enstore knows, see section 8.4 *enstore pnfs*, in particular the `--xref` option of the `enstore pnfs` command.

5.5 Encp Command Options

In this section, we've placed a bomb in front of any option that should be used with utmost care; these options, if misused, can adversely affect not only your jobs, but those of others, as well. We've placed a pointing finger in front of options that, if misused, may adversely affect your own job, but not others' jobs.

💣	--age-time <AGE_TIME>	Specifies the time period, in minutes, after which the priority is eligible to change from the initial job priority. We recommend that you don't set this, just use the default (which is "never").
☞	--array-size <ARRAY_SIZE)	Sets the number of buffers in the array. If --threaded is specified but this option is not, array-size defaults to 3. If this is used without --threaded, this value

		becomes 1 and is ignored. Changing this value for multi-threaded transfers may increase transfer speed.
☞	--buffer-size <BUFFER_SIZE>	Sets the number of bytes of data to transfer at one time (default is 256k). Increasing this value may increase transfer speed. This value must remain lower than the available memory.
☞	--bypass-filesystem-max-filesize-check	Disables the check to protect against the user reading from Enstore a file larger than the maximum size file the local filesystem supports. Use this switch with care.
	--data-access-layer	Turns on special status printing; output has standardized format whether error occurred or not.
💣	--delayed-dismout <DELAY>	Specifies time period in minutes to delay dismount of volume. Use this to tell Enstore: “More work is coming for the volume, don’t dismount the volume too quickly once the current transfer is completed.”
💣	--delpri <DELPRI>	Changes the initial job priority by specified value after a period given by the age-time switch. We recommend that you don’t set this, just use the default (1).
☞	--direct-io	Uses direct I/O for disk access on supporting file systems[1]. Generally, direct I/O makes disk access slower. But when the size of the read/write buffer is made large enough, say, 64Mb or larger, direct I/O is faster because of the skipped memory-to-memory copy.

1. Direct I/O is not universally supported; some filesystems, versions of filesystems, kernels, etc. do not support it. If this doesn’t work for you, contact an enstore admin, and communicate your kernel, library versions, filesystem and filesystem version.

--ecrc	(stands for Enstore crc) This can be used when reading from Enstore. After a file is written to disk, this causes Enstore to reread the disk copy of the file and recalculate the checksum on it.
--ephemeral[1]	This option creates a temporary file family of name “ephemeral”, and copies files to this ephemeral file family on storage media in the order specified. Overrides file family tag in /pnfs destination directory.
--file-family <FILE_FAMILY>	This is used to write data on volumes assigned to specified file family. Overrides file family tag in /pnfs destination directory. (Footnote for --ephemeral applies here, too.)
--help	Displays the list of options for encp.
--map-size <MMAP_SIZE>	The amount of data to map from the file to local memory at one time in bytes (default is 96Mb); use with --mmap-io.
--mmap-io	Uses memory-mapped I/O for disk access on supporting file systems (see the *Enstore Glossary* for an explanation). Make sure you have read and write permissions on the file.
--no-crc	Tells encp to bypass the crc[2] on the local file. (For the minor performance enhancement that this affords, you lose both the encp crc and the one performed by the mover; we discourage use of this option.)
--priority <PRIORITY>	Sets the initial job priority to the specified integer value. We recommend that you don’t set this, just use the default.
--threaded	Multithreads the actual data transfer.

1. The options --ephemeral and --file-family require care when used so that tapes do not get mounted in a way that causes improper and/or inefficient tape usage. Beware of run-away scripts!
2. CRC stands for Cyclic Redundancy Check, a type of checksum.

--usage	Displays information about the **`encp`** options.
--verbose <LEVEL>	Changes the amount of information printed about the transfer; provide an integer value. Default is 0. Larger integer numbers provide more "verbosity". Largest meaningful number may change as development continues.
--version	Displays encp version information.
--pnfs-is-automounted	Typically, users should not automount pnfs. If you do, you can specify this option. It alerts encp to retry errors due to known OS automounting problems. Do not use this in non-automounted cases; it can slow the setup of the transfer.

☞ If you feel compelled to set `--priority`, `--delpri` or `--age-time`, please email *enstore-admin@fnal.gov* first with an explanation, as the defaults should work in almost all cases and changing them may affect other users. Priority goes in strict number sequence, where a higher number means higher priority. Note that Enstore's selection of which file to transfer at a given time uses a much more complicated algorithm than simple priority, however. See Chapter 10: *Job Priority and Queue Management*.

Chapter 6: Copying Directory Structures with Ensync

6.1 About Ensync

Ensync is a wrapper for encp that allows you to copy the contents of an entire directory structure to Enstore via a single command. It is intended for uploading files to Enstore only, not downloading from it. Ensync makes a call to encp to handle each file transfer, and the transfers are done serially. The ensync program is intended for smaller experiments lacking the resources to create their own scripts to do this sort of thing, and without the high data volume of the large experiments.

Ensync works similarly to **`rsync -R`** in that it recursively copies files down a directory hierarchy, which **`encp`** cannot do. It behaves in much the same way that **`rcp`** does, but has more features and uses the rsync remote-update protocol to greatly speed up file transfers when the destination file already exists. It creates an area in PNFS namespace structurally similar to that from which the user is copying. It copies files/directories from a user's local disk to this space. If a file already exists in Enstore with the same relative path/filename as one on the user's disk, the user's file does not get copied; the preexisting Enstore file stays as is.

6.2 Ensync Command Syntax

The command takes two arguments, the "from" and the "to" directories or files. The syntax is:

```
% ensync /<path_to>/.../<local_dir> /pnfs/.../<dest_dir>
```

There are no options to specify. A few notes:

- Symbolic links work if the target is under the same mount point as the link.
- Hard links are not kept. A new copy for each link would get two separate, identical files.
- Ensync tranfers one file at a time, it does not transfer them in parallel.

Chapter 7: Overview of the Enstore Servers

In this chapter we describe the software modules that act as Enstore servers and the libraries with which they interact. The servers include:

- File Clerk (FC)
- Volume Clerk (VC)
- Library Manager (LM)
- Mover (MV)
- Media Changer (MC)
- Configuration Server (CS)

All of the above-listed servers must be running in order for data reads and writes to succeed.

The Enstore monitoring framework includes:

- Inquisitor
- Alarm Server (AS)
- Log Server (LS)
- Event Relay (ER)
- Monitor Server
- Accounting Server
- Drivestat Server

Typically, data transfer can still take place even if any of these monitoring systems is down.

7.1 File Clerk

The File Clerk (FC) is a server that tracks files in the system. It manages a database of metadata for each data file in the Enstore system. The metadata includes the file's name, its unique identifier (the bit file ID, or bfid, that the FC itself assigns to each new file), the volume on which it resides, and so on. You can get information on specific files using the **`enstore info`** command; see section 8.1 *enstore info*.

7.2 Volume Clerk

The Volume Clerk (VC) is a server that stores and administers storage volume (tape) information. You can get information on specific volumes using the **`enstore info`** command; see section 8.1 *enstore info*.

7.3 Library Manager

A Library in Enstore is comprised of both the physical media and a robot arm used to mount the media in attached drives. An Enstore library is typically called a robot. A library/robot interfaces to software that controls the robot arm (the Media Changer, see section 7.5). Each library can contain a variety of media types and employ different types of media drives.

A Virtual Library (VL) is a subset of an Enstore library. It can contain one and only one type of media. A Library Manager (LM) is a server which is bound to a single Virtual Library (VL), and controls what happens within that VL. We speak of bound "LM-VL pairs". An LM receives requests for file reads and writes from the user, stores these unassigned requests in a queue, prioritizes them, and dispatches the requests to a Mover for actual data transfer to and from its VL.

There may be many LM-VL pairs in an Enstore system. There may be more than one LM-VL pair for each media type, but not vice-versa. For example, given an STK Powderhorn library holding 20, 60 and 200 GB media, Enstore would need to divide it into at least three LM-VL pairs.

You can get information on specific library managers using the **`enstore library`** command; see section 8.2 *enstore library*.

7.4 Mover

A Mover (MV) is a process responsible for efficient data transfer between the **encp** process and a single, assigned media drive in a library (robot). The Mover receives instructions from a Library Manager (LM) on how to satisfy the users' requests. The Mover sends instructions to the Media Changer (MC) (described in section 9.5) that services the Mover's assigned drive in order to get the proper volume mounted.

A mover can be configured to serve multiple LMs[1]. Allowing flexible LM assignment has two benefits:

- First, since a virtual library (an LM-VL pair) handles only one type of media, a drive which can handle multiple types of media (e.g., different capacity media) can be shared by multiple LM-VL pairs without a static partitioning of the system.
- Secondly, suppose user groups A and B want to share the capacity of a VL, in which half the tapes belong to group A and the other half to group B. You want to guarantee that groups A and B each get one third of the tape drives, and that the last third is shared. To do this, your administrator can configure the Movers to partition resources in the VL, and assign an LM to each type of use.

7.5 Media Changer

The Media Changer (MC) mounts and dismounts the media into and out of drives according to requests from the Movers. One MC can serve multiple drives and thus multiple VLs (the image in section 7.4 *Mover* shows an MC associated with only one drive). When the drives are in the robot, the MC is the interface to the robotic software.

7.6 Configuration Server

The Configuration Server (CS) maintains and distributes the information about Enstore system configuration, such as the location and parameters of each Enstore component and/or server. At startup, each server asks the CS for the information it needs (e.g., the location of any other server with which it must communicate). New configurations can be loaded into the CS without disturbing the current running system.

7.7 Inquisitor

The Inquisitor monitors the Enstore servers, obtains information from them, and creates reports at regular intervals that can be viewed on the web under `http://hppc.fnal.gov/enstore/`. See section 7.10 *Event Relay* for

1. The media types governed by the LMs must be supported by the Mover's assigned drive.

an illustration of an Inquisitor task. The reports created by the Inquisitor include **Enstore Server Status** (section 9.4), **Encp History** (section 9.10), **Enstore Configuration** (section 9.11), and **Enstore Log Files** (section 9.13).

☞ If the Inquisitor goes down, the **System-At-A-Glance** web page (described in section 9.3) indicates this by a red ball next to *Inquisitor.* In this case, data can still be transferred via **encp**, however, the information on the reports mentioned above doesn't continue to update!

7.8 Alarm Server

The Alarm Server (AS) maintains a record of alarms raised by other servers, and creates a report available online and described in section 9.12 *Enstore Active Alarms*. Since Enstore attempts error recovery whenever possible, it is expected that raised alarms will need human intervention to correct the problem. The AS compares each newly raised alarm with the previously raised ones (it raises a counter) in order to prevent raising the same alarm more than once. Alarm output can be configured to be sent in email messages, to a web page, and so on for notification.

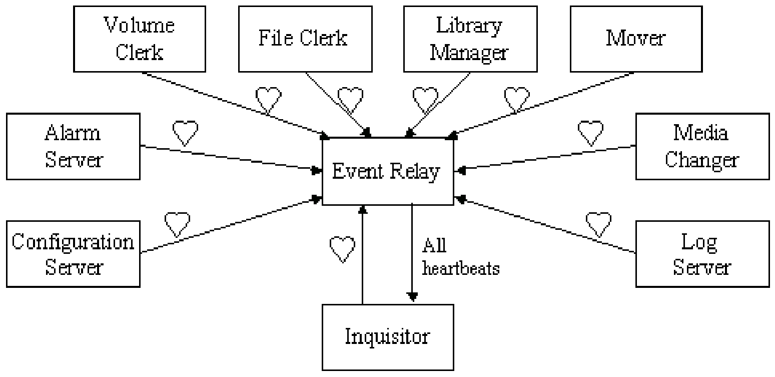
7.9 Log Server

The Log Server (LS) receives messages from other processes and logs them into formatted log files available online and described in section 9.13 *Enstore Log Files*. These messages are transactional records. Log files are labeled by date. Every night at midnight, the currently opened log file gets closed and another one is opened. Logs are backed up to tape.

7.10 Event Relay

The Event Relay (ER) is a server that forwards messages based on subscription. All the Enstore servers send messages to the ER. Any server may "subscribe" to the ER in order to have messages of particular types forwarded to it.

For example, the ER periodically receives "I'm alive" messages (called *heartbeats*) from the other servers in the system. The Inquisitor (section 7.7) subscribes to the heartbeat messages, so the ER forwards these messages to it. This is illustrated in the image below:



☞ If the ER goes down (indicated by a red ball next to *Event Relay* on the **System-At-A-Glance** web page described in section 9.3), the information on **Enstore Server Status** (see section 9.4) and the other web pages described in Chapter 9: *Monitoring Enstore on the Web* may not be accurate.

7.11 Monitor Server

The Monitor Server (MS) is available for investigating network-related problems. It attempts to mimic the communication between **encp**, the corresponding library manager, and the mover. To initiate a test of this kind, you must use the `enstore monitor` command, see section 8.3 *enstore monitor*.

7.12 Accounting Server

The accounting server is a front-end to a Postgres SQL database, and at this time is not intended for use by end users. It maintains statistical information on a running system. Such information is not essential to operations, however it can be used by admins to analyze the performance and utilization of the system for purposes of troubleshooting and future planning.

7.13 Drivestat Server

Drivestat server maintains statistical information of the drives. This is used to update the “ingest rate” plots. As with the accounting server, such information is not essential to operations, however it can be used to analyze the performance and utilization of the system for purposes of troubleshooting and future planning.

7.14 Info Server

The Information Server is a read-only server that maintains detailed file and volume information. You can access this information for particular files and volumes using the **`enstore info`** command and its various options (see section 8.1 *enstore info*).

Chapter 8: Enstore Commands

Enstore provides commands that allow you to communicate with various components of the system. The basic syntax of all Enstore commands is

```
% enstore <command> [--option [argument] ...]
```

All options start with a double dash (`--`).

8.1 enstore info

As of encp v3_2, this command replaces **`enstore file`** and **`enstore vol`**. The developers may remove these latter two commands in future versions of **encp**.

This command communicates with the File Clerk (see section 7.1 *File Clerk*) and the Volume Clerk (see section 7.2 *Volume Clerk*). It returns information about a specified file or files on a specified volume.

Syntax:

```
% enstore info [--option [argument] ... ]
```

Options:

-h, --help

Prints the options (i.e., prints this message). Example:

$ enstore info --help

```
    Usage:
           info [ -h --bfid= --help --list= --ls-active= --usage ]

           --bfid <BFID>         get info of a file
           --gvol <VOLUME_NAME>  get info of a volume in human readable time
                                 format
       -h, --help                print this message
           --just <VOLUME_NAME>  used with --pvols to list problem
           --list <VOLUME_NAME>  list the files in a volume
           --ls-active <VOLUME_NAME>  list active files in a volume
           --ls-sg-count         list all sg counts
           --pvols               list all problem volumes
           --show-bad            list all bad files
           --usage               print short help message
           --vol <VOLUME_NAME>   get info of a volume
           --vols                list all volumes
```

--bfid <BFID>

Returns information (metadata) about the file corresponding to the specified bfid.

You can get the bfid of a file from the **enstore pnfs --bfid <FILE_NAME>** command (section 8.4 *enstore pnfs*); get the filename from searching PNFS namespace.

Example:

$ enstore info --bfid CDMS105770745000000

```
    {'bfid': 'CDMS105770745000000',
     'complete_crc': 1191066979L,
     'deleted': 'no',
     'drive': 'stkenmvr7a:/dev/rmt/tps0d1n:4560000022',
     'external_label': 'VO3222',
     'location_cookie': '0000_000000000_0005661',
     'pnfs_mapname': '',
     'pnfs_name0': '/pnfs/fs/usr/test/xyz/srmtest/ar017983.0001phys_10',
     'pnfsid': '000500000000000000190EA8',
     'pnfsvid': '',
     'sanity_cookie': (65536L, 3203712884L),
     'size': 197354833L}
```

--gvol <VOLUME_NAME>

This is just like **`enstore info --vol <VOLUME_NAME>`**, except that this one prints human-readable time fields (e.g., "declared", "first_access" and "last_access" fields). Example:

`$ enstore info --gvol VO3332`

```
{'blocksize': 131072,
 'capacity_bytes': 64424509440L,
 'declared': 'Wed Jan 16 16:13:57 2002',
 'eod_cookie': '0000_000000000_0000044',
 'external_label': 'VO3332',
 'first_access': 'Fri May 10 12:59:35 2002',
 'last_access': 'Mon Oct 27 22:35:45 2003',
 'library': '9940',
 'media_type': '9940',
 'non_del_files': 43,
 'remaining_bytes': 1785262080L,
 'sum_mounts': 234,
 'sum_rd_access': 213,
 'sum_rd_err': 0,
 'sum_wr_access': 43,
 'sum_wr_err': 0,
 'system_inhibit': ['none', 'full'],
 'user_inhibit': ['none', 'none'],
 'volume_family': 'cms.objy_data_files.cpio_odc',
 'wrapper': 'cpio_odc'}
```

--just

Used with **`--pvols`** to list problem. See **`enstore info --pvols`**.

--list <VOLUME_NAME>

Lists the files in the specified volume with their volume name, bfid, size, location (file number) on volume, delete flag, and the original filename in pnfs.

You can get the volume name from the **`enstore pnfs`** command, using either **`--xref`** or **`--layer`** (section 8.4 *enstore pnfs*), or from the "external_label" field of the **`enstore info --bfid <BFID>`** command (shown above).

The **`enstore info --list <VOLUME_NAME>`** is an alias for this command.

Example:

`$ enstore info --list VO3222`

```
label  bfid                 size   location_cookie        delflag
original_name

VO3222 CDMS106503213600000 983803 0000_000000000_0011536 deleted
/pnfs/fs/usr/eagle/dcache-tests/274.dcache_page_p_27750
```

(This shows one of many lines appearing in the real output, and is reformatted to two lines for readability.)

--ls-active <VOLUME_NAME>

Lists active files in a volume.

You can get the volume name from the **`enstore pnfs`** command, using either **`--xref`** or **`--layer`** (section 8.4 *enstore pnfs*), or from the “external_label” field of the **`enstore info --bfid <BFID>`** command (shown above).

Example:

```
$ enstore info --ls-active VO3222

    /pnfs/fs/usr/eagle/dcache-tests/101.dcache_page_a_24401
    /pnfs/fs/usr/eagle/dcache-tests/101.dcache_page_24401
    /pnfs/fs/usr/test/stress-test/myfile1
    /pnfs/fs/usr/test/stress-test/myfile3
    /pnfs/fs/usr/test/stress-test/file128m-11
    ...
```

--ls-sg-count <VOLUME_NAME>

Lists allocated tape counts by library and by storage group. If “storage group” has value “none”, the negative number under “allocated” gives the number of tapes that are available in the robot, but not yet assigned to a storage group.

Example:

```
$ enstore info --ls-sg-count VO3332

         library    storage group  allocated
    ========================================
           ...
           9940            ktev        189
            9940            lqcd        150
            9940       miniboone        132
            9940           minos        109
            9940            none        -13
            9940         patriot         20
            9940            sdss        608
            9940            test         28
            9940          theory         70
         CD-9940B            cms        129
            ...
```

--pvols [--just <VOLUME_1> <VOLUME_2> ...]

Without **`--just`**, this lists all problem volumes. With **`--just`** followed by a space-separated list of volume names, it lists only the problem volumes among the given list.

The columns returned are: volume name, primary status, primary status time, secondary status, secondary status time. (The time fields are relatively new; not all volumes will display them.)

Example:

`$ enstore info --pvols`

```
==== readonly
LEGL10              none          *     readonly 0913-1540
LEGL98              none          *     readonly 0819-2329
...
==== full
...
VO4845              none          *         full         *
VO4846              none 1023-1032          full         *
VO4847              none          *         full         *
VO4848              none          *         full         *
VO4849              none          *         full         *
VO4850              none          *         full 1016-2315
VO4851              none          *         full 1017-0409
...
$ enstore volume --pvols --just VO3332
(no sample output available)
```

--show-bad

Lists all files that are currently unavailable due to media problems. When a tape problem is discovered, the tape is sent to the vendor for file recovery. In the interim, a cloned tape is made available to users, with the bad files marked. This command option lets you list the bad files. The output lists the tape number, BFID, file size in bites, and pnfs path of file.

Example:

`$ enstore info --show-bad`

```
...
VO0053 CDMS105770745000000 95530315
 /pnfs/fs/usr/xyz/my_data/2004-4/.bad.F000xyz43_0000.mdaq.root
...
```

We show only one output line, and it is displayed on two lines for readability. Notice the "`.bad.`" at the front of the filename; this is how the bad files are marked.

--usage

Prints short help message. Example:

`$ enstore info --usage`

```
Usage:
       info [ -h --bfid= --help --list= --ls-active= --usage ]
```

--vol <VOLUME_NAME>

Returns detailed information about specified volume

Example:

```
$ enstore info --vol VO3332
```

```
    {'blocksize': 131072,
     'capacity_bytes': 64424509440L,
     'declared': 1011219237.849051,
     'eod_cookie': '0000_000000000_0000044',
     'external_label': 'VO3332',
     'first_access': 1021053575.259737,
     'last_access': 1067315745.238969,
     'library': '9940',
     'media_type': '9940',
     'non_del_files': 43,
     'remaining_bytes': 1785262080L,
     'sum_mounts': 234,
     'sum_rd_access': 213,
     'sum_rd_err': 0,
     'sum_wr_access': 43,
     'sum_wr_err': 0,
     'system_inhibit': ['none', 'full'],
     'user_inhibit': ['none', 'none'],
     'volume_family': 'cms.objy_data_files.cpio_odc',
     'wrapper': 'cpio_odc'}
```

--vols

Lists all volumes with their available space, the system inhibits, the library, the volume family (period-separated concatenation of storage group, file family and file family wrapper) and any comments.

Example:

```
$ enstore info --vols
```

```
    label    avail.  system_inhibit   library  vol_family             comment
    ...
    VO0053   1.19GB  (none  full )    eagle    cms.objy_data_files.cpio_odc
    VO0054   0.51GB  (none  full )    eagle    cms.objy_data_files.cpio_odc
    VO0055   0.17GB  (none  full )    eagle    theory.theory-canopy-C.cpio_odc
    VO0056   0.65GB  (none  full )    eagle    theory.theory-canopy-D.cpio_odc
    ...
```

8.2 enstore library

This command communicates with the Library Manager (see section 7.3 *Library Manager*). You can use it to get information pertaining to a particular Library Manager. Use the online monitoring pages (see Chapter 9: *Monitoring Enstore on the Web*) to find the library manager of interest.

Syntax:

```
% enstore library [--option [argument] ... ] <library>
```

The **`<library>`** argument is required except when using the **`--help`** option; the “**`.library_manager`**” portion of the library name is optional.

Options:

-h, --help

Prints this message (i.e., prints the options). Example:

`$ enstore library --help`

```
Usage:
       library [OPTIONS]... library

       --get-asserts <library> print sorted lists of pending volume asserts
       --get-queue <HOST_NAME>  print queue submitted from the specified host.
                              If empty string specified, print the whole queue
       --get-suspect-vols    print suspect volume list
       --get-work-sorted     print sorted lists of pending and active requests
   -h, --help                prints this messge
       --usage               prints short help message
```

--get-asserts <LIBRARY>

Prints sorted lists of pending volume asserts for specified library. Example:

`$ enstore library --get-asserts 9940.library_manager`

```
Pending assert requests: 0
Active assert requests: 0
{'status': ('ok', None)}
```

--get-queue <HOST_NAME> <LIBRARY>

Prints queue submitted from the specified encp client host. Both arguments are required. If quoted empty string is specified for host name, it prints the whole queue (for all hosts). Examples:

```
$ enstore library --get-queue stkensrv3 9940.library_manager
```

```
    Pending write requests
    Active requests
    Pending read requests:  0
    Pending write requests:  2
    Active read requests:  0
    Active write requests:  0
    {'status': ('ok', None)}
```

The top two lines tell us that there are no pending or active transfers involving stkensrv3 for the 9940 library manager. The 4th line tells us there are 2 pending write requests for this library manager from hosts other than stkensrv3.

If all hosts are specified (the next example), the command returns the fields: host name, library manager, username (of encp request), input filename, and output filename for each pending and/or active request (3 shown here), and ends with a summary:

```
$ enstore library --get-queue "" 9940.library_manager
```

```
    Active requests
    fnsimu2 9940.library_manager lixn
       /pnfs/btev/geant2003/xiaonan/dstar_xiaonan_1.evt.gz
       /scr/bphys6/lixn/dstar_xiaonan_1.evt.gz M 9944
    fsgi01 9940.library_manager rschultz
       /usr/bdms/rschultz/fl_066_uplsr7/fl_ed_066_uplsr7.ldhi
       /pnfs/BDMS/lens/fl_066_uplsr7/fl_ed_0663
    fnsfh 9940.library_manager minfarm
       /export/stage02_minos/C00040259_0000.tdaq.root
       /pnfs/minos/caldet_reco/tdaq_data/2002-09/C0004027
    Pending read requests:  0
    Pending write requests:  0
    Active read requests:  1
    Active write requests:  2
    {'status': ('ok', None)}
```

--get-suspect-vols <LIBRARY>

Prints suspect volume list for specified library manager. Example:

```
$ enstore library --get-suspect-vols 9940.library_manager
```

```
    [{'movers': ['994071.mover'], 'external_label': 'VO4523',
    'time': 1067290586.907726}, {'movers': ['994051.mover', '994061.mover', ']
```

--get-work-sorted <LIBRARY>

Prints sorted lists of pending and active requests. It sorts by queue. Example:

```
$ enstore library --get-work-sorted 9940.library_manager
```

```
    {'write_queue': [], 'read_queue': [], 'admin_queue': []}
    [{'status': ('ok', None), 'vc': {'status': ('ok', None),
    'declared': 1011741604.130481, 'si_time': [1041612783.99499, 0], 'blocksiz]
```

8.3 enstore monitor

This command communicates with the Monitor Server (see Chapter 9: *Monitoring Enstore on the Web*) to get network speed information.

On machines with an `enstore.conf` file (see Appendix A: *Network Control*), the **`enstore monitor`** command uses the routing already established there. If **`enstore monitor`** set up its own, it would interfere with the routes currently in use.

Syntax:

```
% enstore monitor [--option [argument] ...]
```

-h, --help

Prints this message (i.e., prints the options). Example:

```
$ enstore monitor -h
Usage:
       monitor [ -h --help --host= --usage --verbose= ]

   -h, --help               prints this messge
       --host <HOSTIP>      selects a single host
       --usage              prints short help message
       --verbose <VERBOSE>  print out information.
```

--host [HOST_NAME or IP_ADDRESS]

Returns network rate for the specified host (Enstore node). If you don’t specify host, it runs the command for all hosts. Example below shows results for a single host. Example:

```
$ enstore monitor --host stkensrv3
```

```
Trying stkensrv3.fnal.gov
Network rate measured at 11.33 MB/S recieving and 11.1 MB/S sending.
```

--verbose <INTEGER_VALUE>

This command is used to help find and fix network problems. It prints detailed information about actions taken. The higher the number you give as an argument, the more info displayed. Example:

`$ enstore monitor --host stkensrv3 --verbose 20`

```
6  Tue  Oct  28  10:48:13  2003  msc  called  with  args:  ['monitor',  '--host',
'stkensrv3', '--verbose=20']
13 Tue Oct 28 10:48:13 2003 Get monitor_server config info from server
Trying stkensrv0.fnal.gov
13 Tue Oct 28 10:48:13 2003 Get None config info from server
13 Tue Oct 28 10:48:13 2003 Get None config info from server
13 Tue Oct 28 10:48:13 2003 Get log_server config info from server
13 Tue Oct 28 10:48:13 2003 Get log_server config info from server
13 Tue Oct 28 10:48:13 2003 Get None config info from server
13 Tue Oct 28 10:48:13 2003 Get alarm_server config info from server
...
10 Tue Oct 28 10:48:14 2003 Connecting to monitor server.
10 Tue Oct 28 10:48:14 2003 Obtaining error status for data socket.
10 Tue Oct 28 10:48:15 2003 Get the final dialog rate information.
Network rate measured at 11.34 MB/S recieving and 11.23 MB/S sending.
```

8.4 enstore pnfs

Enstore has a **`pnfs`** command that allows you to perform a variety of pnfs manipulations, as listed in the option table below. Off-site users cannot mount `/pnfs`, and therefore cannot run this command.

Syntax:

`% enstore pnfs [--option [argument] ... ]`

--help

List the options for the **`enstore pnfs`** command. Example:

```
% enstore pnfs --help

    Usage:
           pnfs [OPTIONS]...

           --bfid <FILENAME>     lists the bit file id for file
           --cat <FILENAME> [LAYER]  see --layer
           --file-family [FILE_FAMILY]  gets file family tag, default; sets file
                                 family tag, optional
           --file-family-width [FILE_FAMILY_WIDTH]  gets file family width tag,
                                 default; sets file family tag, optional
           --file-family-wrapper [FILE_FAMILY_WRAPPER]  gets file family width tag,
                                 default; sets file family tag, optional
           --filesize <FILE>     print out real filesize
       -h, --help                prints this messge
           --info <FILENAME>     see --xref
           --layer <FILENAME> [LAYER]  lists the layer of the file
           --library [LIBRARY]   gets library tag, default; sets library tag,
                                 optional
           --tag <TAG> [DIRECTORY]  lists the tag of the directory
           --tagchmod <PERMISSIONS> <TAG>  changes the permissions for the tag; use
                                 UNIX chmod style permissions
           --tagchown <OWNER> <TAG>  changes the ownership for the tag; OWNER can
                                 be 'owner' or 'owner.group'
           --tags [DIRECTORY]    lists tag values and permissions
           --usage               prints short help message
           --xref <FILENAME>     lists the cross reference data for file
```

--bfid <FILE_NAME>

Returns the BFID of the file; select file name to specify from within pnfs space and use relative/absolute path as needed.

Example:

```
$ enstore pnfs --bfid /pnfs/mist/zuu/100MB_002

    WAMS104102942800000
```

--cat <PATH_TO_FILE> [LAYER]

--cat is an alias for --layer; see --layer.

--file-family

Prints the file family name associated with the current pnfs directory. Example:

```
$ enstore pnfs --file-family

    dcache
```

--file-family-width
 	
Prints the file family width associated with the current pnfs directory. Example:
 	
`$ enstore pnfs --file-family-width`
 	
`1`

--file-family-wrapper Prints the file family wrapper associated with the current pnfs directory. Example:
 	
`$ enstore pnfs --file-family-wrapper`
 	
`cpio_odc`

--filesize <PATH_TO_FILE>
 	
Prints the real filesize in bytes; useful for files of size greater than (2G-1) bytes, since PNFS stores file size as 1 in this case. Example:
 	
`$ enstore pnfs --filesize a01`
 	
`24198`

--info <PATH_TO_FILE> Prints information about the file, this is an alias for the --xref option. See --xref.

--layer <PATH-TO-FILE> <LAYER>

Prints information about the file. Layer 0 is used internally by pnfs and it can't be viewed. Layer 1, the default, gives the file's BFID. Layer 2 is used by dCache. Layers 3, 5, 6, 7 are not currently used. Layer 4 produces output equivalent to --xref, but returns info without field labels.

The option --cat is an alias for this option.

Examples:

Layer 1 gives BFID (default):

$ enstore pnfs --layer a01

```
CDMS105889726300000
```

$ enstore pnfs --layer a01 1

```
CDMS105889726300000
```

Layer 2 is used for dCache:

$ enstore pnfs --layer a01 2

```
2,0,0,0.0,0.0
:c=1:d15ef6a3;l=554423;
w-fcdfdata018-1
```

The file has a version1 crc of c=1:d15ef6a3, it has a length l=554423, and it is in pool w-fcdfdata018-1.

$ enstore pnfs --layer a01 2

```
2,0,0,0.0,0.0
:
```

Layer 4 gives --xref output (see --xref):

$ enstore pnfs --layer a01 4

```
VO3222
0000_000000000_0006264
24198
dcache
/pnfs/fs/usr/test/xyz/srmtest/test_1_1/a01

000500000000000000191030

CDMS105889726300000
stkenmvr5a:/dev/rmt/tps3d1n:4560000022
```

--tags [DIRECTORY]

List the tag values of specified PNFS directory (if no directory argument, it lists tags for current working directory (cwd or pwd)). Example:

```
$ pwd
    /pnfs/test/xyz/srmtest/test_1_1
$ enstore pnfs --tags
    .(tag)(file_family) = dcache
    .(tag)(file_family_width) = 1
    .(tag)(file_family_wrapper) = cpio_odc
    .(tag)(library) = 9940
    .(tag)(storage_group) = test
    -rw-rw-r--   11 root     sys            6 Jul 26  2001
                     /pnfs/test/xyz/srmtest/test_1_1/.(tag)(file_family)
    -rw-rw-r--   11 root     sys            1 May  5  2000
                     /pnfs/test/xyz/srmtest/test_1_1/.(tag)(file_family_width)
    -rw-rw-r--   11 root     sys            8 May  5  2000
                     /pnfs/test/xyz/srmtest/test_1_1/.(tag)(file_family_wrapper)
    -rw-rw-r--   11 root     sys            4 Jul  3 10:59
                     /pnfs/test/xyz/srmtest/test_1_1/.(tag)(library)
    -rw-r--r--   11 root     sys            4 Jul 26  2001
                     /pnfs/test/xyz/srmtest/test_1_1/.(tag)(storage_group)
```

(minor reformatting done to enhance readability)

--xref <FILE_NAME>

List cross-reference information (metadata) for specified file. (--info is an alias for --xref.) The information includes:

- volume: storage media volume label
- location cookie: file position on tape (the number of the file on tape)
- size: file size in bytes
- file family: file family
- original name: original name in `/pnfs` before any move/copy command issued; i.e., the destination filename given in the **`encp`** command used to copy the file to Enstore
- map file: obsolete, but some older files may have a value here
- pnfsid file: unique id for the file as assigned by PNFS
- pnfsid map: obsolete, but some older files may have a value here
- bfid: unique id for the file as assigned by Enstore (matches layer 1)
- origdrive: id of drive used when file was written to media (files generated prior to 10/2000, encp v2.5 or earlier, will not have a value here)
- crc: CRC of the file (appears for files after 10/2003, using encp v3_1 or greater)

Example:

```
$ enstore pnfs --xref a01

    volume: VO3222
    location_cookie: 0000_000000000_0006264
    size: 24198
    file_family: dcache
    original_name: /pnfs/fs/usr/test/xyz/srmtest/test_1_1/a01
    map_file:
    pnfsid_file: 000500000000000000191030
    pnfsid_map:
    bfid: CDMS105889726300000
    origdrive: stkenmvr5a:/dev/rmt/tps3d1n:4560000022
    crc: unknown
```

--library

Returns the value of the library tag (the virtual library associated with files in the directory) for the current pnfs directory. Example:

```
$ enstore pnfs --library

    9940
```

8.5 enstore file (deprecated)

This command has been deprecated for users (not for admins) as of encp v3_2, and (along with **`enstore volume`**) replaced with **`enstore info`** (see section 8.1 *enstore info*).

This command communicates with the File Clerk (see section 7.1 *File Clerk*). It returns information about a specified file or files on a specified volume.

Syntax:

```
% enstore file [--option [argument] ... ]
```

Options:

-h, --help

Prints the options (i.e., prints this message). Example:

```
$ enstore file --help
```

```
    Usage:
           file [ -h --bfid= --help --list= --ls-active= --usage ]

           --bfid <BFID>         get info of a file
       -h, --help                print this message
           --list <VOLUME_NAME>  list the files in a volume
           --ls-active <VOLUME_NAME>  list active files in a volume
           --usage               print short help message
```

--bfid <BFID>

Returns information (metadata) about the file corresponding to the specified bfid.

You can get the bfid of a file from the **`enstore pnfs --bfid <FILE_NAME>`** command (section 8.4 *enstore pnfs*); get the filename from searching PNFS namespace.

Example:

```
$ enstore file --bfid CDMS105770745000000
```

```
    {'bfid': 'CDMS105770745000000',
     'complete_crc': 1191066979L,
     'deleted': 'no',
     'drive': 'stkenmvr7a:/dev/rmt/tps0d1n:4560000022',
     'external_label': 'VO3222',
     'location_cookie': '0000_000000000_0005661',
     'pnfs_mapname': '',
     'pnfs_name0': '/pnfs/fs/usr/test/xyz/srmtest/ar017983.0001phys_10',
     'pnfsid': '000500000000000000190EA8',
     'pnfsvid': '',
     'sanity_cookie': (65536L, 3203712884L),
     'size': 197354833L}
```

--list <VOLUME_NAME>

Lists the files in the specified volume with their volume name, bfid, size, location (file number) on volume, delete flag, and the original filename in pnfs.

You can get the volume name from the **`enstore pnfs`** command, using either **`--xref`** or **`--layer`** (section 8.4 *enstore pnfs*), or from the "external_label" field of the **`enstore file --bfid <BFID>`** command (shown above).

The **`enstore volume --list <VOLUME_NAME>`** is an alias for this command.

Example:

```
$ enstore file --list VO3222
```

```
label  bfid                size   location_cookie        delflag
original_name

VO3222 CDMS106503213600000 983803 0000_000000000_0011536 deleted
/pnfs/fs/usr/eagle/dcache-tests/274.dcache_page_p_27750
```

(This shows one of many lines appearing in the real output, and is reformatted to two lines for readability.)

--ls-active <VOLUME_NAME>

Lists active files in a volume.

You can get the volume name from the **`enstore pnfs`** command, using either **`--xref`** or **`--layer`** (section 8.4 *enstore pnfs*), or from the "external_label" field of the **`enstore file --bfid <BFID>`** command (shown above).

Example:

```
$ enstore file --ls-active VO3222
```

```
/pnfs/fs/usr/eagle/dcache-tests/101.dcache_page_a_24401
/pnfs/fs/usr/eagle/dcache-tests/101.dcache_page_24401
/pnfs/fs/usr/test/stress-test/myfile1
/pnfs/fs/usr/test/stress-test/myfile3
/pnfs/fs/usr/test/stress-test/file128m-11
...
```

--usage

Prints short help message. Example:

```
$ enstore file --usage
```

```
Usage:
       file [ -h --bfid= --help --list= --ls-active= --usage ]
```

8.6 enstore volume (deprecated)

This command has been deprecated for users (not admins) as of encp v3_2, and replaced (along with **`enstore file`**) with **`enstore info`** (see section 8.1 *enstore info*).

This command communicates with the Volume Clerk (see section 7.2 *Volume Clerk*) to return information on data volumes.

Syntax:

```
% enstore volume [--option [argument] ... ]
```

-h, --help

Prints this message (i.e., prints the options). Example:

```
$ enstore volume --help
```

```
    Usage:
           volume [OPTIONS]...

           --gvol <VOLUME_NAME>  get info of a volume in human readable time
                                 format
       -h, --help                prints this messge
           --just <VOLUME_NAME>  used with --pvols to list problem
           --list <VOLUME_NAME>  list the files in a volume
           --ls-active <VOLUME_NAME>  list active files in a volume
           --ls-sg-count         list all sg counts
           --pvols               list all problem volumes
           --usage               prints short help message
           --vol <VOLUME_NAME>   get info of a volume
           --vols                list all volumes
```

--gvol <VOLUME_NAME>

This is just like **`enstore volume --vol <VOLUME_NAME>`**, except that this one prints human-readable time fields (e.g., “declared”, “first_access” and “last_access” fields). Example:

`$ enstore volume --gvol VO3332`

```
{'blocksize': 131072,
 'capacity_bytes': 64424509440L,
 'declared': 'Wed Jan 16 16:13:57 2002',
 'eod_cookie': '0000_000000000_0000044',
 'external_label': 'VO3332',
 'first_access': 'Fri May 10 12:59:35 2002',
 'last_access': 'Mon Oct 27 22:35:45 2003',
 'library': '9940',
 'media_type': '9940',
 'non_del_files': 43,
 'remaining_bytes': 1785262080L,
 'sum_mounts': 234,
 'sum_rd_access': 213,
 'sum_rd_err': 0,
 'sum_wr_access': 43,
 'sum_wr_err': 0,
 'system_inhibit': ['none', 'full'],
 'user_inhibit': ['none', 'none'],
 'volume_family': 'cms.objy_data_files.cpio_odc',
 'wrapper': 'cpio_odc'}
```

--just

Used with **`--pvols`** to list problem. See **`enstore volume --pvols`**.

--list <VOLUME_NAME>

This is an alias for the **`enstore file --list <VOLUME_NAME>`** command. See section 8.5 *enstore file (deprecated)*.

--ls-active <VOLUME_NAME>

Lists original file names of active files in a volume. Example:

`$ enstore volume --ls-active VO3332`

```
/pnfs/cms/UserFederation/data/jetmet_production/data/Collections/jm_Hit601_g12
5_UCSD/jm02_qqh120_ll/EVD0.jet0102.DB
/pnfs/cms/UserFederation/data/jetmet_production/data/TAssoc/jm_2x1033PUjm602_T
kMu_g125_UCSD/jm02_hlt15-20/EVD11.jet0102.DB
/pnfs/cms/UserFederation/data/jetmet_production/data/Digis/jm_2x1033PUjm602_Tk
Mu_g125_UCSD/jm02-hlt0-15/EVD12.jet0102.DB
/pnfs/cms/UserFederation/data/jetmet_production/data/Hits/jm_Hit601_g125_UCSD/
jm02_hlt230-300/EVD12.jet0102.DB
...
```

--ls-sg-count <VOLUME_NAME>

Lists allocated tape counts by library and by storage group. If "storage group" has value "none", the negative number under "allocated" gives the number of tapes that are available in the robot, but not yet assigned to a storage group.

Example:

$ enstore volume --ls-sg-count VO3332

```
        library    storage group  allocated
    ========================================
           ...
           9940            ktev        189
            9940            lqcd        150
            9940      miniboone        132
            9940           minos        109
            9940            none        -13
            9940         patriot         20
            9940            sdss        608
            9940            test         28
            9940          theory         70
        CD-9940B             cms        129
            ...
```

--pvols [--just <VOLUME_1> <VOLUME_2> ...]

Without **`--just`**, this lists all problem volumes. With **`--just`** followed by a space-separated list of volume names, it lists only the problem volumes among the given list.

The columns returned are: volume name, primary status, primary status time, secondary status, secondary status time. (The time fields are relatively new; not all volumes will display them.)

Example:

$ enstore volume --pvols

```
    ==== readonly
    LEGL10             none         *     readonly 0913-1540
    LEGL98             none         *     readonly 0819-2329
    ...
    ==== full
    ...
    VO4845             none         *         full         *
    VO4846             none 1023-1032         full         *
    VO4847             none         *         full         *
    VO4848             none         *         full         *
    VO4849             none         *         full         *
    VO4850             none         *         full 1016-2315
    VO4851             none         *         full 1017-0409
    ...
    $ enstore volume --pvols --just VO3332
    (no sample output available)
```

--vol <VOLUME_NAME>

Returns detailed information about specified volume

Example:

```
$ enstore volume --vol VO3332
```

```
{'blocksize': 131072,
 'capacity_bytes': 64424509440L,
 'declared': 1011219237.849051,
 'eod_cookie': '0000_000000000_0000044',
 'external_label': 'VO3332',
 'first_access': 1021053575.259737,
 'last_access': 1067315745.238969,
 'library': '9940',
 'media_type': '9940',
 'non_del_files': 43,
 'remaining_bytes': 1785262080L,
 'sum_mounts': 234,
 'sum_rd_access': 213,
 'sum_rd_err': 0,
 'sum_wr_access': 43,
 'sum_wr_err': 0,
 'system_inhibit': ['none', 'full'],
 'user_inhibit': ['none', 'none'],
 'volume_family': 'cms.objy_data_files.cpio_odc',
 'wrapper': 'cpio_odc'}
```

--vols

Lists all volumes with their available space, the system inhibits, the library, the volume family (period-separated concatenation of storage group, file family and file family wrapper) and any comments.

Example:

```
$ enstore volume --vols
```

```
label     avail.  system_inhibit   library  vol_family             comment
...
VO0053    1.19GB  (none  full )    eagle    cms.objy_data_files.cpio_odc
VO0054    0.51GB  (none  full )    eagle    cms.objy_data_files.cpio_odc
VO0055    0.17GB  (none  full )    eagle    theory.theory-canopy-C.cpio_odc
VO0056    0.65GB  (none  full )    eagle    theory.theory-canopy-D.cpio_odc
...
```

Chapter 9: Monitoring Enstore on the Web

There are several installed Enstore systems at Fermilab. Currently these include STKEN for general Fermilab users, CDFEN for CDF RunII, and D0EN for D0 RunII. For each Enstore system, a separate but structurally identical series of web pages is available for monitoring the system and any jobs you've submitted to it. The currently implemented websites for Enstore monitoring include:

- `http://www-stken.fnal.gov/enstore/enstore_system.html` for STKEN
- `http://www-cdfen.fnal.gov/enstore/enstore_system.html` for CDFEN
- `http://www-d0en.fnal.gov/enstore/enstore_system.html` for D0EN

We recommend that you bookmark the appropriate one in your browser.

In this section, we briefly describe the format and function of the web pages that are of interest to users, and show you how to navigate them.

The Enstore pages present snapshots of the status of various components of the Enstore system, and the pages are updated and refreshed periodically. The auto-refresh time interval varies from page to page, and does not correspond with the information update interval, which also varies from page to page. See the online help screens for more detailed information.

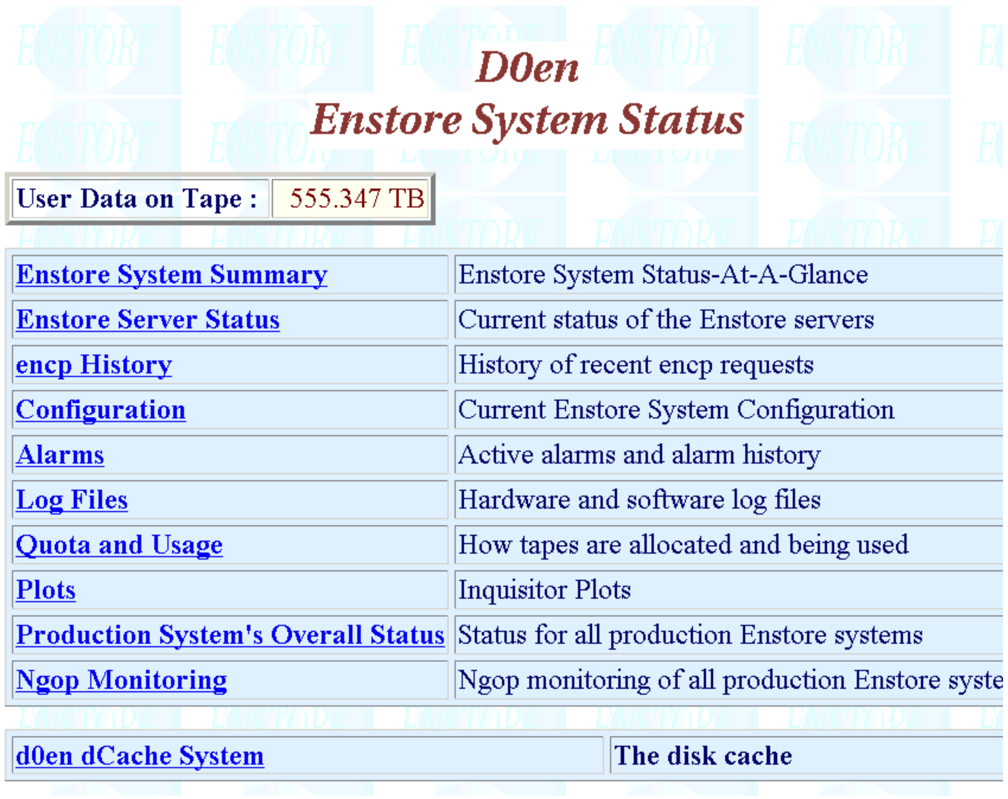
☞ Note for Netscape users: Links on these pages are intended to take you straight to the item of interest, not to the top of the page on which it's found. Due to a Netscape bug, you'll find yourself at the top of the target page. To get to the item of interest, place your cursor in the URL area of the browser and hit **ENTER**.

9.1 Top Page

The top page for monitoring an Enstore system is located at `http://www-<xyz>en.fnal.gov/enstore/` (where `<xyz>` is one of `stk`, `cdf` or `d0`), as given above. This page has two sections, each containing links to other pages.

9.1.1 Enstore System Status Links

The links under the *Enstore System Status* heading lead to status web pages for the Enstore system and its servers, shown here for the D0en system:



D0en
Enstore System Status

User Data on Tape :	555.347 TB

Enstore System Summary	Enstore System Status-At-A-Glance
Enstore Server Status	Current status of the Enstore servers
encp History	History of recent encp requests
Configuration	Current Enstore System Configuration
Alarms	Active alarms and alarm history
Log Files	Hardware and software log files
Quota and Usage	How tapes are allocated and being used
Plots	Inquisitor Plots
Production System's Overall Status	Status for all production Enstore systems
Ngop Monitoring	Ngop monitoring of all production Enstore syste

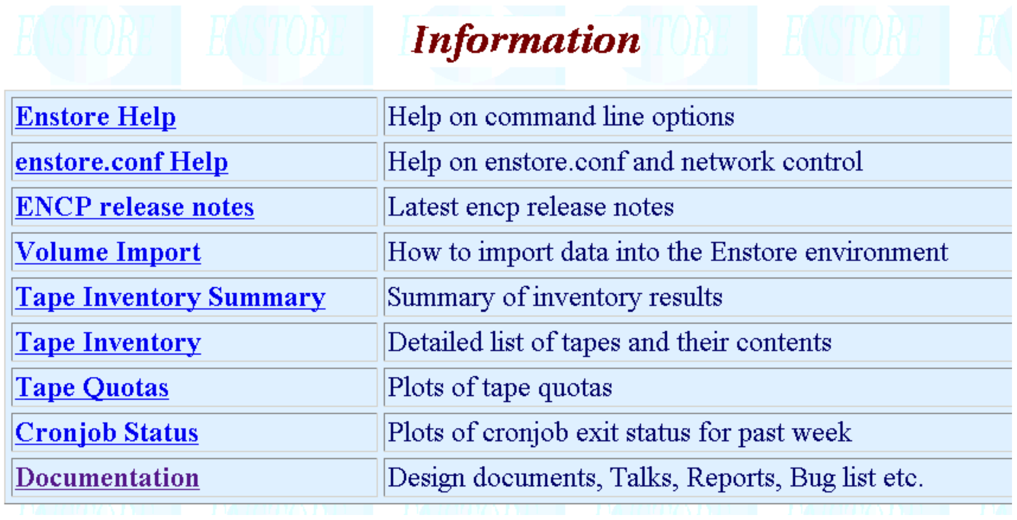
d0en dCache System	The disk cache

The pages to which these links point (with the exceptions of *Quota and Usage* and *Production System's Overall Status*[1]) share a header format, described in section 9.2 *Header Format for Status Pages*.

9.1.2 Information

Under the *Information* header are links for finding help, documentation, and so on.

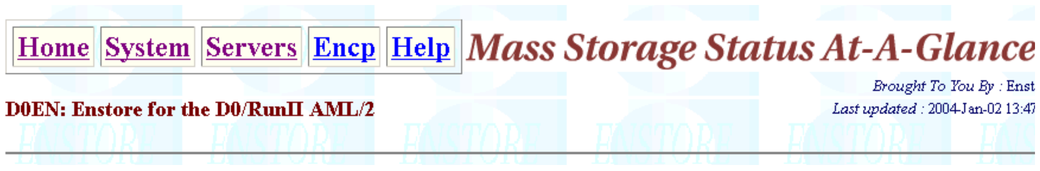
1. *Quota and Usage* links to a text file with no header; and *Production System's Overall Status* has the header elements on the right-hand side only.



Enstore Help	Help on command line options
enstore.conf Help	Help on enstore.conf and network control
ENCP release notes	Latest encp release notes
Volume Import	How to import data into the Enstore environment
Tape Inventory Summary	Summary of inventory results
Tape Inventory	Detailed list of tapes and their contents
Tape Quotas	Plots of tape quotas
Cronjob Status	Plots of cronjob exit status for past week
Documentation	Design documents, Talks, Reports, Bug list etc.

9.2 Header Format for Status Pages

Here we see the header for the **Mass Storage Status At-A-Glance** page (from the link “Enstore System Summary” on the top page):



Home
System
Servers
Encp
Help
Mass Storage Status At-A-Glance
Brought To You By : Enst
D0EN: Enstore for the D0/RunII AML/2
Last updated : 2004-Jan-02 13:47

Header elements:

- In the upper-right corner you’ll find the page title, **Mass Storage Status At-A-Glance**, in this case.
- Underneath the page title is the name of the Enstore server that created this web page (e.g., Enstore), and the date and time that the current page was created. If the time shown here is more than a few minutes earlier than the current time, you should refresh your browser window to get updated information.
- The buttons in the upper-left corner are quick links to different pages:

Home	the top page, described in section 9.1 *Top Page*

System	the **Status At-A-Glance** page, described in section 9.3 *Mass Storage Status-At-A-Glance Page* (the page associated with the “Enstore System Summary” link on the top page; it is the page shown in the above image)
Servers	the **Enstore Server Status** page, described in section 9.4 *Enstore Server Status* (the page associated with the “Enstore Server Status” link on the top page)
Encp	the **Encp History** page, described in section 9.10 *Encp History* (the page associated with the “encp History” link on the top page)
Help	page-specific online help

• Underneath these buttons you’ll find the Enstore system identifier; in the above image, it is *D0EN: Enstore for the D0/RunII AML/2*.

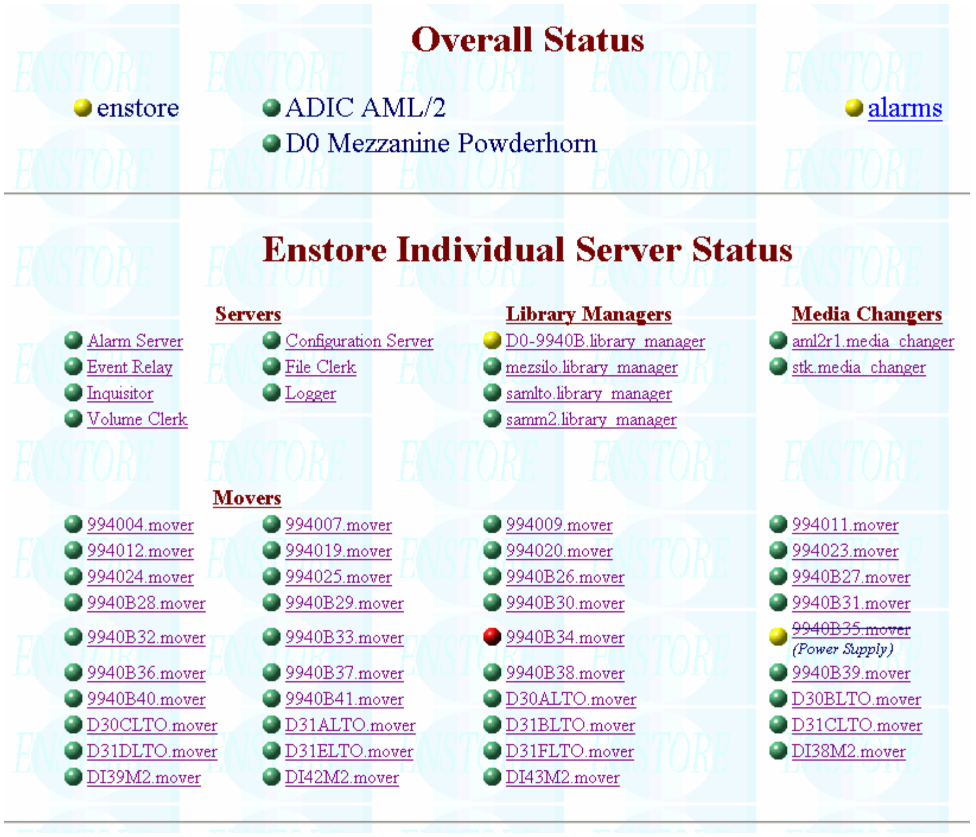
9.3 Mass Storage Status-At-A-Glance Page

What?	The **Status-At-A-Glance** page presents summarized information indicating which parts of the Enstore system are up and working, which parts have problems, which have a scheduled outage, and other system information. It also provides a mapping between Enstore servers and the nodes that run them.
Why?	Start at this page when investigating any possible problem. This page indicates which if any components of your Enstore system are experiencing problems.
How?	To arrive at this page, start at the top page and click “Enstore System Summary”, or click the **SYSTEM** button on any of the pages.

The Enstore components and servers listed on this page are described in Chapter 7: *Overview of the Enstore Servers*.

Page Description

The page is divided into three sections. They list systems and servers, and code them with colored ball icons to indicate their status. The **HELP** button at the top of the page (see section 9.2 *Header Format for Status Pages*) describes the status icons.



Overall Status

enstore	ADIC AML/2	alarms
	D0 Mezzanine Powderhorn	

Enstore Individual Server Status

Servers		Library Managers	Media Changers
Alarm Server	Configuration Server	D0-9940B.library_manager	aml2r1.media_changer
Event Relay	File Clerk	mezsilo.library_manager	stk.media_changer
Inquisitor	Logger	samlto.library_manager	
Volume Clerk		samm2.library_manager	

Movers

994004.mover	994007.mover	994009.mover	994011.mover
994012.mover	994019.mover	994020.mover	994023.mover
994024.mover	994025.mover	9940B26.mover	9940B27.mover
9940B28.mover	9940B29.mover	9940B30.mover	9940B31.mover
9940B32.mover	9940B33.mover	9940B34.mover	~~9940B35.mover~~ *(Power Supply)*
9940B36.mover	9940B37.mover	9940B38.mover	9940B39.mover
9940B40.mover	9940B41.mover	D30ALTO.mover	D30BLTO.mover
D30CLTO.mover	D31ALTO.mover	D31BLTO.mover	D31CLTO.mover
D31DLTO.mover	D31ELTO.mover	D31FLTO.mover	DI38M2.mover
DI39M2.mover	DI42M2.mover	DI43M2.mover	

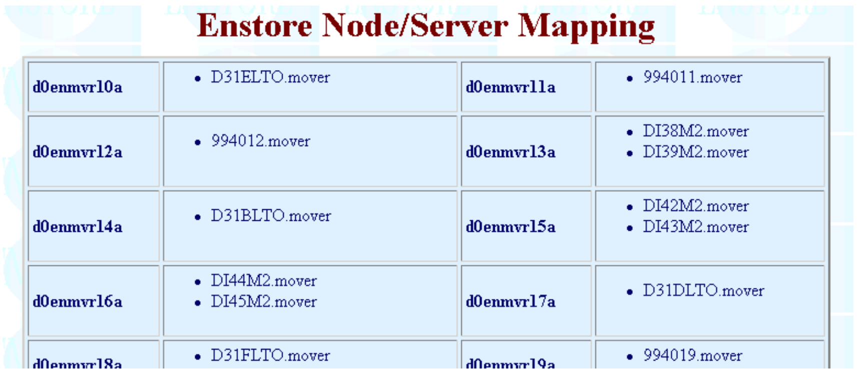
Enstore Overall Status summarizes the status (from left to right) of Enstore as a whole, the tape robots, the network, and alarm components. There is only one link:

The "alarms" link takes you to the **Enstore Active Alarms** page described in section 9.12 *Enstore Active Alarms*.

Enstore Individual Server Status

lists all servers (Chapter 7), library managers (section 7.3), movers (section 7.4), and media changers (section 7.5); includes individual status indicators. Each link in this section takes you to its corresponding server entry on the page described in section 9.4 *Enstore Server Status*.

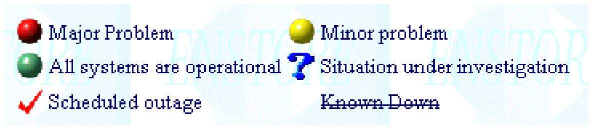
Status indicators do not apply to the third section, which lists the nodes in the Enstore system and the servers that run on each of them (below we show the first few rows of the table). There is no status information.



Enstore Node/Server Mapping

d0enmvr10a	• D31ELTO.mover	d0enmvr11a	• 994011.mover
d0enmvr12a	• 994012.mover	d0enmvr13a	• DI38M2.mover • DI39M2.mover
d0enmvr14a	• D31BLTO.mover	d0enmvr15a	• DI42M2.mover • DI43M2.mover
d0enmvr16a	• DI44M2.mover • DI45M2.mover	d0enmvr17a	• D31DLTO.mover
d0enmvr18a	• D31FLTO.mover	d0enmvr19a	• 994019.mover

There is a legend at the bottom of the page for the status icons which looks like this:



🔴 Major Problem	🟡 Minor problem
🟢 All systems are operational	? Situation under investigation
✓ Scheduled outage	~~Known Down~~

9.4 Enstore Server Status

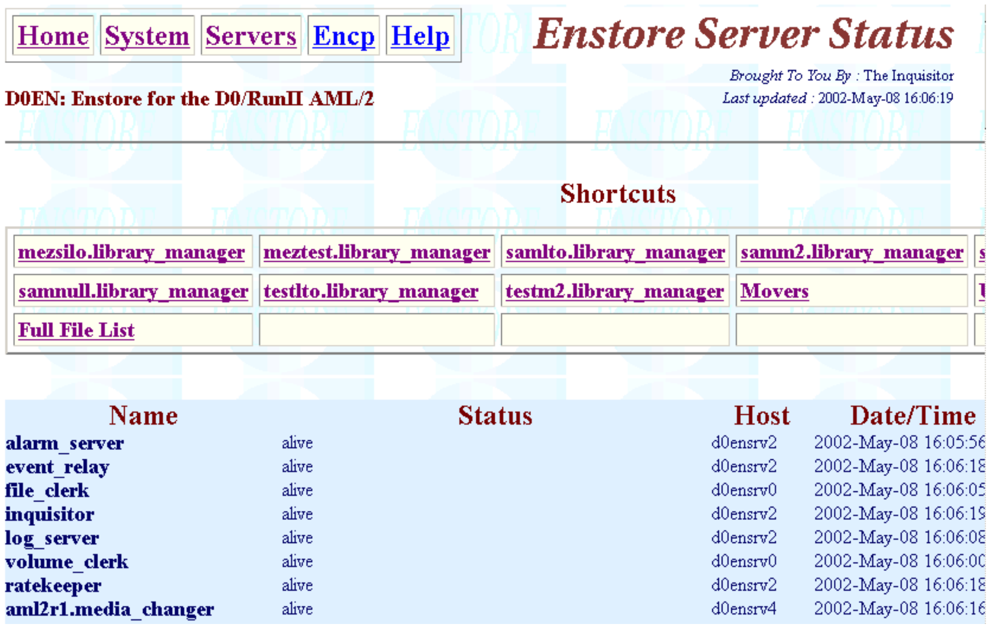
What? As the page title implies, the **Enstore Server Summary** page provides the status of all the Enstore servers included in your system. This includes movers, library managers, and so on. The servers are described in Chapter 7: *Overview of the Enstore Servers*.

Why? Use this page to find out what a particular server is currently doing, and what work it has pending.

How? To arrive at this page, start at the top page and click “Enstore Server Status”, or click the **SERVERS** button on any of the pages.

Page Description

The page is divided into two sections.



Enstore Server Status

Brought To You By : The Inquisitor
Last updated : 2002-May-08 16:06:19

D0EN: Enstore for the D0/RunII AML/2

Shortcuts

mezsilo.library_manager	meztest.library_manager	samlto.library_manager	samm2.library_manager	s
samnull.library_manager	testlto.library_manager	testm2.library_manager	Movers	U
Full File List				

Name	Status	Host	Date/Time
alarm_server	alive	d0ensrv2	2002-May-08 16:05:56
event_relay	alive	d0ensrv2	2002-May-08 16:06:18
file_clerk	alive	d0ensrv0	2002-May-08 16:06:05
inquisitor	alive	d0ensrv2	2002-May-08 16:06:19
log_server	alive	d0ensrv2	2002-May-08 16:06:08
volume_clerk	alive	d0ensrv0	2002-May-08 16:06:00
ratekeeper	alive	d0ensrv2	2002-May-08 16:06:18
aml2r1.media_changer	alive	d0ensrv4	2002-May-08 16:06:16

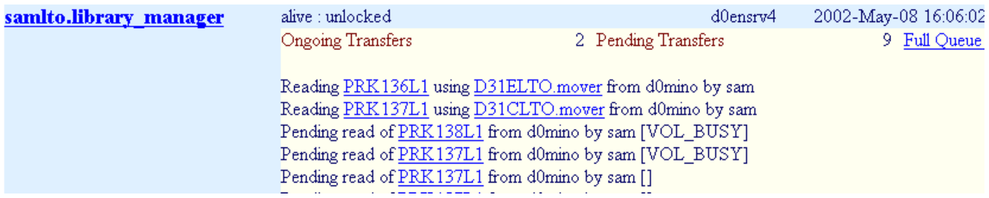
The first section, *Shortcuts*, is simply a compilation of links that point to anchors in the table that comprises the second section. There is also a link labelled “Full File List” which takes you to the **Active File List** page, described in section 9.5 *Active File List* (useful for tracking down the library managers associated with the file you want to investigate if you only know the name of the file).

The second (and main) section is a status table which lists all the servers, and displays the server name, status, host, date/time, and last time alive for each. Some server names and status information in this table have links to pages with more information. We define the statuses below by server type.

This page is updated and refreshed periodically.

Library Managers

- The link on a library manager (LM) name points to the **Library Manager Queues** page for the corresponding LM (see section 9.6 *Library Manager Queues*).
- The link “Full Queue Elements” points to the **Full Library Manager Info** page (see section 9.7 *Full Library Manager Info*).
- The link on a volume name takes you to a text page with the volume’s inventory information (see section).
- The link on a mover name points to the **Movers** page (see section 9.9 *Movers Page*).



samlto.library_manager	alive : unlocked	d0ensrv4	2002-May-08 16:06:02
	Ongoing Transfers 2 Pending Transfers		9 Full Queue

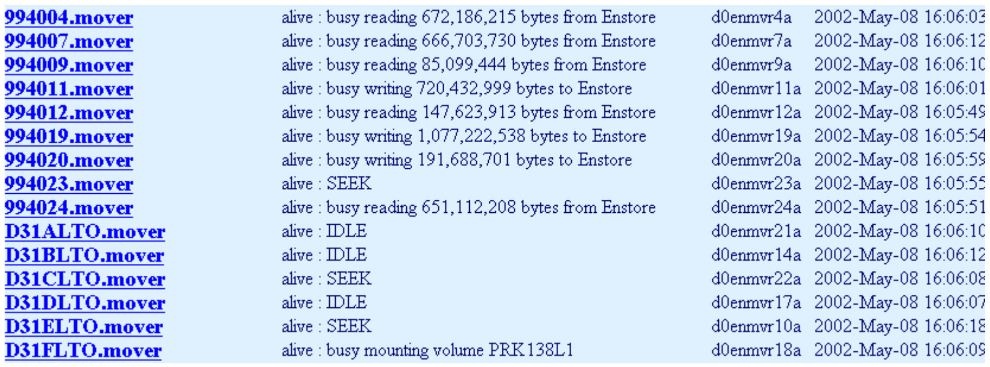
Reading PRK136L1 using D31ELTO.mover from d0mino by sam
Reading PRK137L1 using D31CLTO.mover from d0mino by sam
Pending read of PRK138L1 from d0mino by sam [VOL_BUSY]
Pending read of PRK137L1 from d0mino by sam [VOL_BUSY]
Pending read of PRK137L1 from d0mino by sam []

Statuses for library managers (LM) include:

`alive : unlocked`	LM is working normally
`alive: locked`	LM is rejecting new **encp** requests, but continues to assign jobs already in the pending queue to movers; **encp** does not retry
`alive: nowrite`	LM is locked for write requests
`alive: noread`	LM is locked for read requests
`alive: ignore`	LM is ignoring new **encp** requests (returning “ok” to **encp**), but continues to assign jobs already in the pending queue to movers; **encp** retries internally so user is unaware of the delay
`alive: pause`	LM is ignoring new **encp** requests, and holding pending jobs

Movers

The link on a mover name points to the corresponding mover (MV) on the **Movers** page (described in section 9.9 *Movers Page*).



994004.mover	alive : busy reading 672,186,215 bytes from Enstore	d0enmvr4a	2002-May-08 16:06:03
994007.mover	alive : busy reading 666,703,730 bytes from Enstore	d0enmvr7a	2002-May-08 16:06:12
994009.mover	alive : busy reading 85,099,444 bytes from Enstore	d0enmvr9a	2002-May-08 16:06:10
994011.mover	alive : busy writing 720,432,999 bytes to Enstore	d0enmvr11a	2002-May-08 16:06:01
994012.mover	alive : busy reading 147,623,913 bytes from Enstore	d0enmvr12a	2002-May-08 16:05:49
994019.mover	alive : busy writing 1,077,222,538 bytes to Enstore	d0enmvr19a	2002-May-08 16:05:54
994020.mover	alive : busy writing 191,688,701 bytes to Enstore	d0enmvr20a	2002-May-08 16:05:59
994023.mover	alive : SEEK	d0enmvr23a	2002-May-08 16:05:55
994024.mover	alive : busy reading 651,112,208 bytes from Enstore	d0enmvr24a	2002-May-08 16:05:51
D31ALTO.mover	alive : IDLE	d0enmvr21a	2002-May-08 16:06:10
D31BLTO.mover	alive : IDLE	d0enmvr14a	2002-May-08 16:06:12
D31CLTO.mover	alive : SEEK	d0enmvr22a	2002-May-08 16:06:08
D31DLTO.mover	alive : IDLE	d0enmvr17a	2002-May-08 16:06:07
D31ELTO.mover	alive : SEEK	d0enmvr10a	2002-May-08 16:06:18
D31FLTO.mover	alive : busy mounting volume PRK138L1	d0enmvr18a	2002-May-08 16:06:09

Statuses for movers include:

`alive : IDLE`	MV is idle because there are no jobs to process
`alive : SETUP`	MV is in initial phase of a job, it is setting up a connection with **encp** for a transfer

`alive: busy mounting volume <volname>`	
	MV is waiting for the media changer to finish mounting a tape; the volume name is given
`alive : SEEK`	a tape is mounted, and the correct read or write location on the tape is being located
`alive : busy reading/writing <n> bytes from/to Enstore`	
	MV is reading data from Enstore, or writing data to Enstore; the number of bytes read or written so far is given
`alive : busy dismounting volume <volname>`	
	MV is waiting for media changer to finish dismounting a volume; the volume name is given
`alive : HAVE BOUND volume - IDLE`	
	MV has completed a job but is waiting for a subsequent job for same tape; tape is still in drive
`alive: DRAINING`	
	MV is completing last job before going offline; it will not accept more jobs
`alive : CLEANING`	
	a cleaning tape is in the drive; MV cannot accept more jobs until the cleaning has finished.
`alive : OFFLINE`	
	MV is offline and not accepting jobs (MV name displayed in orange)
`alive : FINISH_WRITE`	
	MV writing is completed and MV is waiting for file and volume metadata to be created.
`alive : ERROR <text>`	
	MV is in an error state described by the text, and cannot accept more jobs (MV name displayed in orange)

Other Servers

Statuses for all the servers except movers and library managers:

`alive`	server is working normally

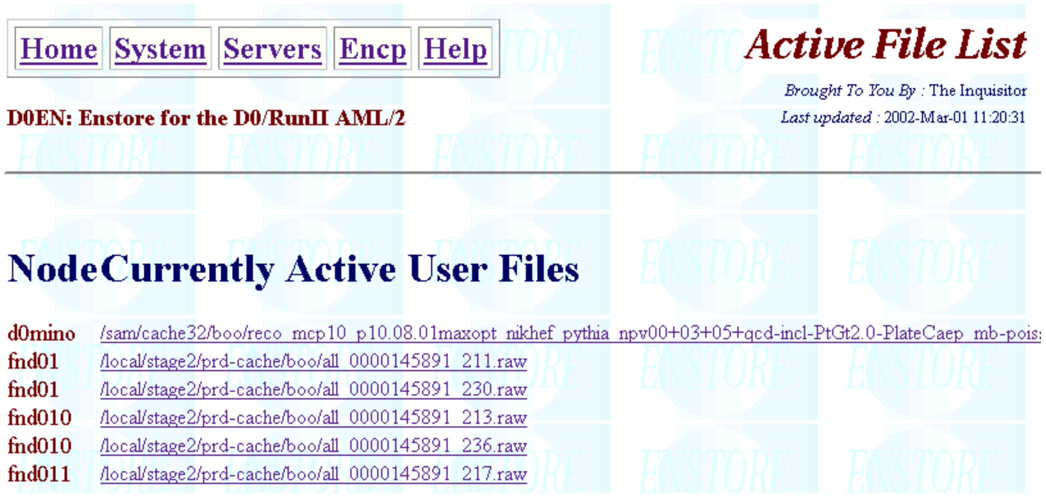
`timed out`	the inquisitor hasn't received the latest "I'm alive" message from the server
`dead`	duration of "timed out" status on server has exceeded configured limit (server name appears in orange)
`not monitoring`	server is known to the enstore system, but is not currently being monitored by the inquisitor (server name is displayed in gray)

9.5 Active File List

What?	The **Active File List** page lists the data files being actively worked on by your Enstore system. The files are listed by user node.
Why?	Use this page to find your file. This is the right starting page for checking on your job if you only know the name of the file you're reading or writing (i.e., you don't know the volume or any Enstore server information). This page has links to pages containing more job-related information.
How?	To arrive at this page, start at the top page and click "Enstore Server Status". Then under *Shortcuts*, click "Full File List".

Page Description

The files are listed by their full path and name. For each file, the node from/to which it is being read/written is also given. This page doesn't distinguish between read and write.



Active File List

Brought To You By : The Inquisitor
Last updated : 2002-Mar-01 11:20:31

D0EN: Enstore for the D0/RunII AML/2

NodeCurrently Active User Files

d0mino	/sam/cache32/boo/reco mcp10 p10.08.01maxopt nikhef pythia npv00+03+05+qcd-incl-PtGt2.0-PlateCaep mb-pois:
fnd01	/local/stage2/prd-cache/boo/all 0000145891 211.raw
fnd01	/local/stage2/prd-cache/boo/all 0000145891 230.raw
fnd010	/local/stage2/prd-cache/boo/all 0000145891 213.raw
fnd010	/local/stage2/prd-cache/boo/all 0000145891 236.raw
fnd011	/local/stage2/prd-cache/boo/all 0000145891 217.raw

Scroll as necessary or do a search to find your file in the list and click on it. This will take you to the **Library Manager Queues** page for the library manager servicing your job; see section 9.6 *Library Manager Queues*.

9.6 Library Manager Queues

What? The **Library Manager Queues** page lists the **encp** jobs that a selected library manager is currently managing or has pending in a queue (**encp** is described in Chapter 5: *Copying Files with Encp*). Movers in states other than busy or IDLE are listed at the bottom of the page (mover statuses are described in section 9.4 *Enstore Server Status*).

Why? Use this page to find out the status of a particular library manager's read and/or write queue(s) once you know which LM is servicing your job. You can find the status of your job, and its priority relative to other jobs in the queue. From this page you can click links to get full details on the processing of your file and on the volume associated with your file.

How? To arrive at this page, follow this string of links starting at the top page. Click "Enstore Server Status", then:

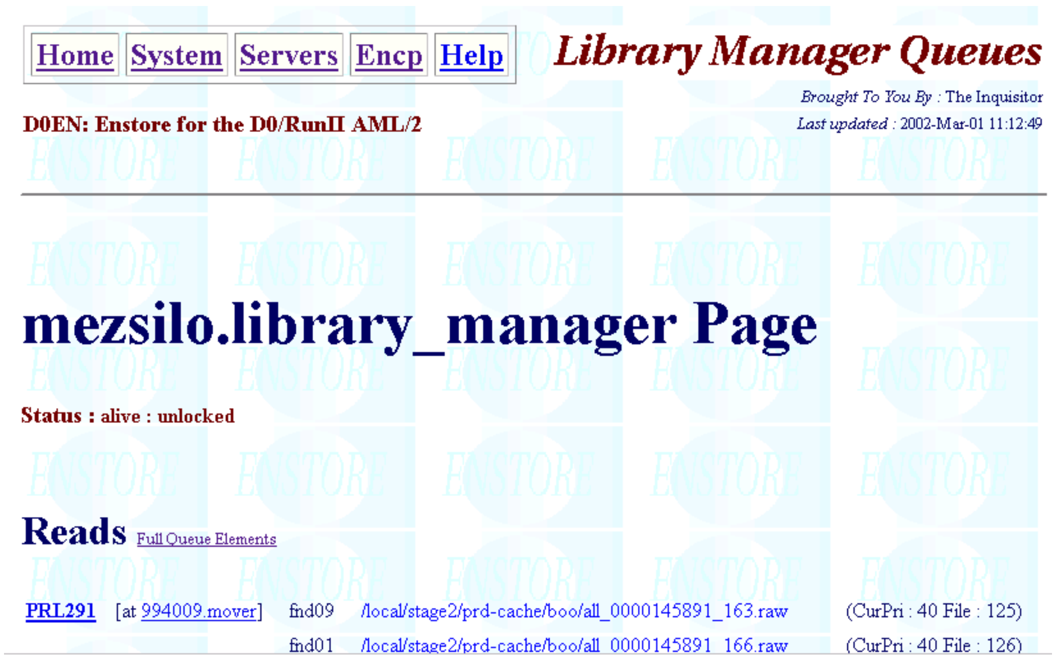
- If you know the filename but not the library manager, then under *Shortcuts*, click "Full File List". Click on your file of interest. This will take you to the **Library Manager Queues** page for the appropriate library manager.
- If you know the library manager, you can click directly on the link in the second section of the **Enstore Server Status** page, instead.

9.6.1 Suspect Volumes

(no write-up yet)

9.6.2 File Reads

For the *Reads*, files are listed by volume. For each volume in use, the page lists the mover servicing it. Each **encp** job is listed on a separate line. The line lists the host to which the file is to be copied, the last 70 or so bytes of the filename (filenames can get quite long), the file's current priority in the queue, and the file's position on the tape. The files are ordered in the queue by priority.



Library Manager Queues

Brought To You By : The Inquisitor
Last updated : 2002-Mar-01 11:12:49

D0EN: Enstore for the D0/RunII AML/2

mezsilo.library_manager Page

Status : alive : unlocked

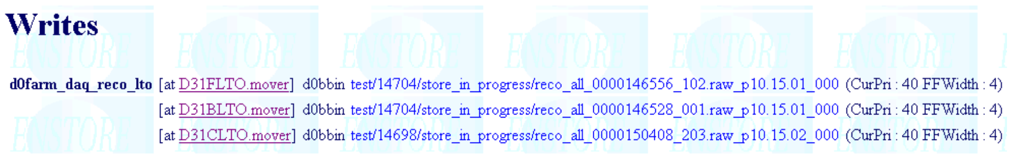
Reads Full Queue Elements

```
PRL291  [at 994009.mover]  fnd09  /local/stage2/prd-cache/boo/all_0000145891_163.raw  (CurPri : 40 File : 125)
                           fnd01  /local/stage2/prd-cache/boo/all_0000145891_166.raw  (CurPri : 40 File : 126)
```

To get full information on the library manager's processing, click *Full Queue Elements* next to *Reads* to arrive at the *Full Library Manager Info* page (described in section 9.7 *Full Library Manager Info*). To get full information on a volume being read, click the volume id at the top-left of the queue containing your file. This takes you to the text inventory page for that volume (described in section).

9.6.3 File Writes

Under *Writes*, this page lists write jobs by file family. A mover is listed after the file family for a file only if the file is currently being worked on.



```
Writes
d0farm_daq_reco_lto [at D31FLTO.mover] d0bbin test/14704/store_in_progress/reco_all_0000146556_102.raw_p10.15.01_000 (CurPri : 40 FFWidth : 4)
                    [at D31BLTO.mover] d0bbin test/14704/store_in_progress/reco_all_0000146528_001.raw_p10.15.01_000 (CurPri : 40 FFWidth : 4)
                    [at D31CLTO.mover] d0bbin test/14698/store_in_progress/reco_all_0000150408_203.raw_p10.15.02_000 (CurPri : 40 FFWidth : 4)
```

Each file in the write queue appears on a separate line. Each line lists several pieces of information: the host from which the file is to be copied, the last several bytes of the filename (filenames can get quite long), and the current priority and file family width.

The mover name provides a link to the **Movers** page, described in section 9.9 *Movers Page*.

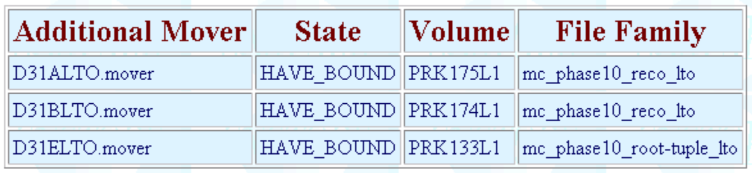
Job Processing and File Family Width

Normally, the number of WRITE jobs running per file family can equal but not exceed the file family width (see section 1.4.2 *File Family Width*). If a READ job is running on a tape that is not marked full, this also counts against the width.

☞ But note: Even if the number of current jobs equals the width, it is possible for a new READ job to start on a tape that's not full (if the tape is marked full, the width is not an issue and the READ job can start anyway) since the width is checked only when assigning WRITE jobs; thus temporarily, the width may be exceeded. Any pending WRITE job must wait until the the number of jobs that count against the width drops below the width value.

9.6.4 Additional Movers

Underneath this information, there may be a table listing additional movers.



Additional Mover	State	Volume	File Family
D31ALTO.mover	HAVE_BOUND	PRK175L1	mc_phase10_reco_lto
D31BLTO.mover	HAVE_BOUND	PRK174L1	mc_phase10_reco_lto
D31ELTO.mover	HAVE_BOUND	PRK133L1	mc_phase10_root-tuple_lto

Movers that are in any of the following states are listed here (see section 9.4 *Enstore Server Status* for status descriptions):

- CLEANING
- DISMOUNT_WAIT
- ERROR
- HAVE_BOUND
- OFFLINE

Movers not listed anywhere on the page may be assumed to be IDLE, i.e., waiting for a job.

9.7 Full Library Manager Info

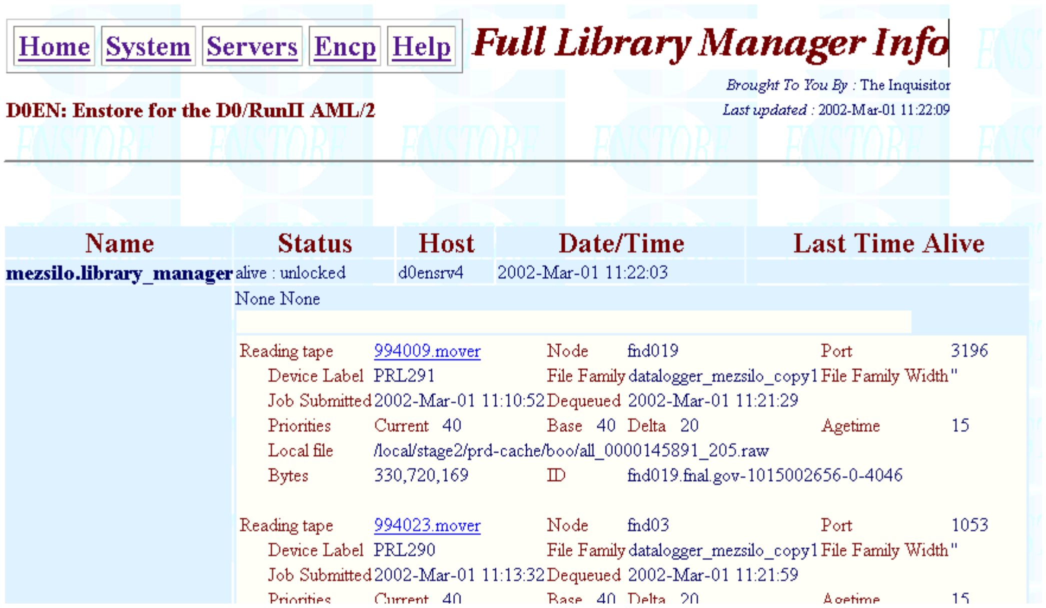
What? The **Full Library Manager Info** page displays the job parameters for each file in a given library manager's current READ and WRITE queues (e.g., local file name, local node, file family, volume ID, priority, etc.).

Why? Use this page to find the status of a READ or WRITE job, e.g., the file's position in the queue, how long it's been in a queue, when it was "dequeued" (i.e., when processing started), and other details about how Enstore is processing it.

How? To arrive at this page, follow this string of links starting at the top page. Click “Enstore Server Status”, find the library manager you want, and click “Full Queue Elements”.

If you don’t know which LM you want but you know the file, take this route. On the **Enstore Server Status** page under *Shortcuts*, click “Full File List”. On the **Active File List** page, click on your file of interest. This will take you to the **Library Manager Queues** page for the appropriate library manager. Here, click “Full Queue Elements” next to *Reads* to come to the **Full Library Manager Info** page. On this page, you can scroll down and locate your file.

Page Description



Full Library Manager Info

Brought To You By : The Inquisitor
Last updated : 2002-Mar-01 11:22:09

D0EN: Enstore for the D0/RunII AML/2

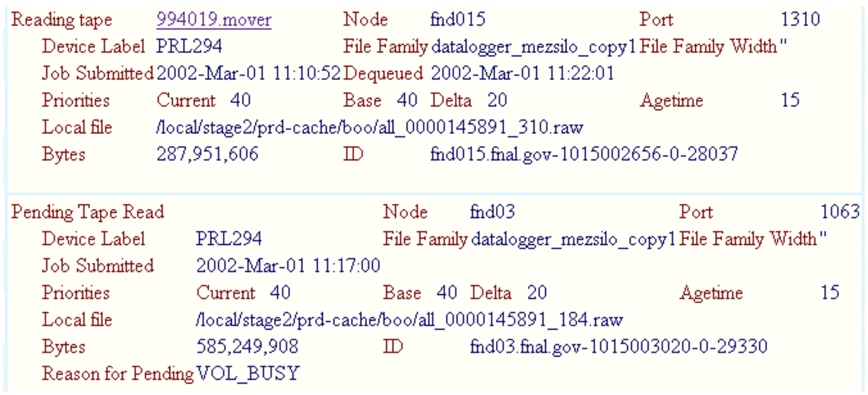
Name	Status	Host	Date/Time	Last Time Alive
mezsilo.library_manager	alive : unlocked	d0ensrv4	2002-Mar-01 11:22:03	

```
None None

Reading tape    994009.mover         Node        fnd019                    Port              3196
  Device Label  PRL291               File Family datalogger_mezsilo_copy1  File Family Width "
  Job Submitted 2002-Mar-01 11:10:52 Dequeued    2002-Mar-01 11:21:29
  Priorities    Current  40          Base  40    Delta  20                 Agetime           15
  Local file    /local/stage2/prd-cache/boo/all_0000145891_205.raw
  Bytes         330,720,169          ID          fnd019.fnal.gov-1015002656-0-4046

Reading tape    994023.mover         Node        fnd03                     Port              1053
  Device Label  PRL290               File Family datalogger_mezsilo_copy1  File Family Width "
  Job Submitted 2002-Mar-01 11:13:32 Dequeued    2002-Mar-01 11:21:59
  Priorities    Current  40          Base  40    Delta  20                 Agetime           15
```

Your file will appear as one of two types of entries on this page: one type for files being worked on, and another for files pending in the queue.



```
Reading tape      994019.mover      Node       fnd015               Port              1310
    Device Label  PRL294            File Family datalogger_mezsilo_copy1 File Family Width "
    Job Submitted 2002-Mar-01 11:10:52 Dequeued 2002-Mar-01 11:22:01
    Priorities    Current  40       Base  40  Delta  20             Agetime           15
    Local file    /local/stage2/prd-cache/boo/all_0000145891_310.raw
    Bytes         287,951,606       ID         fnd015.fnal.gov-1015002656-0-28037

Pending Tape Read                        Node       fnd03                Port              1063
    Device Label       PRL294            File Family datalogger_mezsilo_copy1 File Family Width "
    Job Submitted      2002-Mar-01 11:17:00
    Priorities         Current  40       Base  40  Delta  20             Agetime           15
    Local file         /local/stage2/prd-cache/boo/all_0000145891_184.raw
    Bytes              585,249,908       ID         fnd03.fnal.gov-1015003020-0-29330
    Reason for Pending VOL_BUSY
```

For those files being worked on, the mover name is given, and it provides a link to the **Movers** page, described in section 9.9 *Movers Page*.

9.8 Tape Inventory Page (Text)

There are a couple of pages that present volume inventory information. One is straight text, discussed in this section. The other page is dynamically generated HTML; see section 9.17 *Tape Inventory Page (Dynamic HTML)* . The formats of both pages are similar.

What? For each volume declared to your Enstore system, there is a page that presents volume inventory information in straight text format. The page gets updated periodically; be aware that it may not reflect the most recent information.

Why? Use this page to find out details of the storage of your file(s) on a volume, to see how full a tape is, or to check the inhibits.

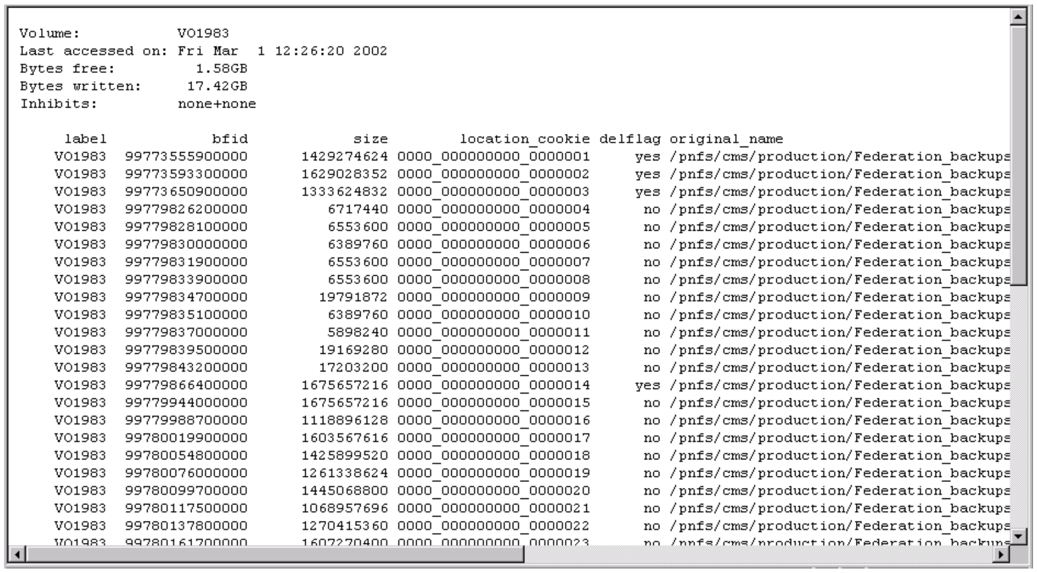
How? *If you only know the filename:* To arrive at the text web page, follow this string of links starting at the top page: click “Enstore Server Status”; then under *Shortcuts*, click “Full File List”. Click on your file of interest. This will take you to the **Library Manager Queues** page for the appropriate library manager. Find your file, and click the corresponding volume ID to come to the inventory page for that volume.

If you know the volume name: To arrive at the text web page, follow this string of links starting at the top page: click “Enstore Server Status”; then look for the volume name listed with one of the active library managers, and click on the volume.

Page Description

At the top of the volume inventory (text) page, you'll find the volume ID, the last accessed date, the number of bytes free, the number of bytes written, and the inhibits (described below).

The volume inventory contains a line for each file on the volume, listed in location order. In addition to the tape label, this page lists the bfid, size, location_cookie, delflag, and original_name (the name given in the **encp** command used to write it). Scroll down to the bottom of the page to find information for the tape volume itself.



```
Volume:           VO1983
Last accessed on: Fri Mar  1 12:26:20 2002
Bytes free:         1.58GB
Bytes written:     17.42GB
Inhibits:         none+none

     label           bfid           size       location_cookie delflag original_name
    VO1983  99773555900000      1429274624 0000_000000000_0000001     yes /pnfs/cms/production/Federation_backups
    VO1983  99773593300000      1629028352 0000_000000000_0000002     yes /pnfs/cms/production/Federation_backups
    VO1983  99773650900000      1333624832 0000_000000000_0000003     yes /pnfs/cms/production/Federation_backups
    VO1983  99779826200000         6717440 0000_000000000_0000004      no /pnfs/cms/production/Federation_backups
    VO1983  99779828100000         6553600 0000_000000000_0000005      no /pnfs/cms/production/Federation_backups
    VO1983  99779830000000         6389760 0000_000000000_0000006      no /pnfs/cms/production/Federation_backups
    VO1983  99779831900000         6553600 0000_000000000_0000007      no /pnfs/cms/production/Federation_backups
    VO1983  99779833900000         6553600 0000_000000000_0000008      no /pnfs/cms/production/Federation_backups
    VO1983  99779834700000        19791872 0000_000000000_0000009      no /pnfs/cms/production/Federation_backups
    VO1983  99779835100000         6389760 0000_000000000_0000010      no /pnfs/cms/production/Federation_backups
    VO1983  99779837000000         5898240 0000_000000000_0000011      no /pnfs/cms/production/Federation_backups
    VO1983  99779839500000        19169280 0000_000000000_0000012      no /pnfs/cms/production/Federation_backups
    VO1983  99779843200000        17203200 0000_000000000_0000013      no /pnfs/cms/production/Federation_backups
    VO1983  99779866400000      1675657216 0000_000000000_0000014     yes /pnfs/cms/production/Federation_backups
    VO1983  99779944000000      1675657216 0000_000000000_0000015      no /pnfs/cms/production/Federation_backups
    VO1983  99779988700000      1118896128 0000_000000000_0000016      no /pnfs/cms/production/Federation_backups
    VO1983  99780019900000      1603567616 0000_000000000_0000017      no /pnfs/cms/production/Federation_backups
    VO1983  99780054800000      1425899520 0000_000000000_0000018      no /pnfs/cms/production/Federation_backups
    VO1983  99780076000000      1261338624 0000_000000000_0000019      no /pnfs/cms/production/Federation_backups
    VO1983  99780099700000      1445068800 0000_000000000_0000020      no /pnfs/cms/production/Federation_backups
    VO1983  99780117500000      1068957696 0000_000000000_0000021      no /pnfs/cms/production/Federation_backups
    VO1983  99780137800000      1270415360 0000_000000000_0000022      no /pnfs/cms/production/Federation_backups
    VO1983  99780161700000      1607270400 0000_000000000_0000023      no /pnfs/cms/production/Federation_backups
```

Inhibits

The inhibits are listed on the page in the format `system_inhibit[0] - system_inhibit[1]`.

`system_inhibit[0]` can take any of the following values:

none	the normal state (no inhibits)
READONLY	volume is read-only
DELETED	volume has been deleted, but admins can still restore the metadata if the volume has not been reused.
NOACCESS	no access allowed (set by system to prevent further access to volume on which it found an error; once the problem is resolved, operator must clear the NOACCESS state)
NOTALLOW	no access allowed (set manually by the operator to prevent access to volume)

`system_inhibit[1]` can take any of the following values:

none	the normal state (no inhibits)
full	volume is full
migrated	files have been migrated to another tape

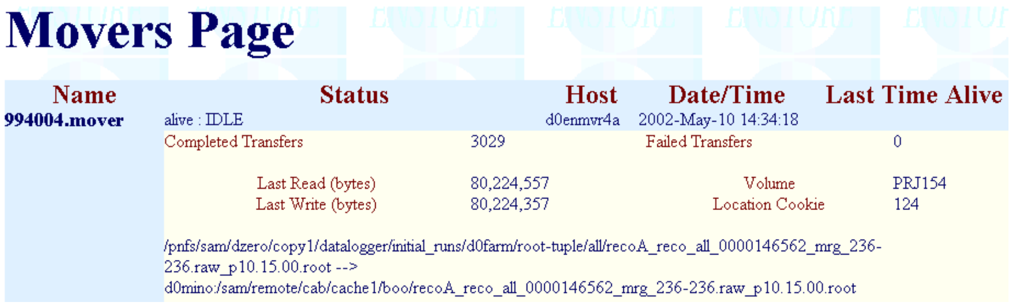
9.9 Movers Page

What?	The **Movers** page displays the current status of all the movers. (The mover statuses are described in section 9.4 *Enstore Server Status.*)
Why?	Use this page to see how far into a job a mover is, or to check other job details related to the mover, e.g., what volume is being used for your job.
How?	There are several paths to arrive at this page. The two easiest and most common are: • On the top page click "Enstore Server Status". Click on a mover. • On the top page click "Enstore Server Status". Choose a library manager to get to the **Library Manager Queues** page, then click on a mover.

When you click on a specific mover, you are brought to the entry for that mover on the **Movers** page.

Page Description

This web page shows the most recent known state of all of the movers in the Enstore system. The first image (below) shows the field headings. The online help page provides a detailed description of the fields.



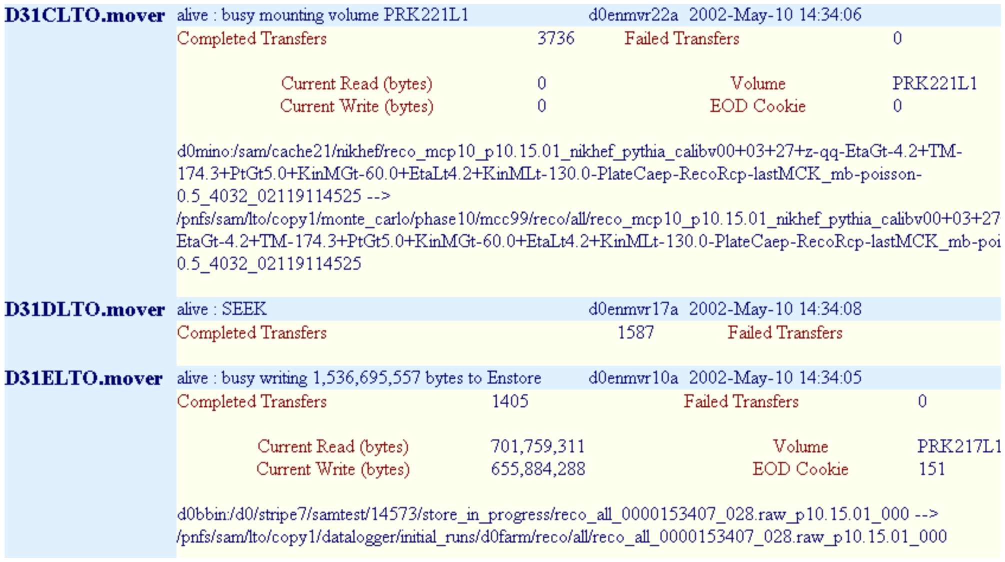
Movers Page

Name	Status	Host	Date/Time	Last Time Alive
994004.mover	alive : IDLE	d0enmvr4a	2002-May-10 14:34:18	

Completed Transfers	3029	Failed Transfers	0
Last Read (bytes)	80,224,557	Volume	PRJ154
Last Write (bytes)	80,224,357	Location Cookie	124

/pnfs/sam/dzero/copy1/datalogger/initial_runs/d0farm/root-tuple/all/recoA_reco_all_0000146562_mrg_236-236.raw_p10.15.00.root -->
d0mino:/sam/remote/cab/cache1/boo/recoA_reco_all_0000146562_mrg_236-236.raw_p10.15.00.root

This next image shows movers that are busy mounting, seeking and writing tapes. The pnfs and user filenames are given as appropriate:



D31CLTO.mover alive : busy mounting volume PRK221L1 d0enmvr22a 2002-May-10 14:34:06

Completed Transfers	3736	Failed Transfers	0
Current Read (bytes)	0	Volume	PRK221L1
Current Write (bytes)	0	EOD Cookie	0

d0mino:/sam/cache21/nikhef/reco_mcp10_p10.15.01_nikhef_pythia_calibv00+03+27+z-qq-EtaGt-4.2+TM-174.3+PtGt5.0+KinMGt-60.0+EtaLt4.2+KinMLt-130.0-PlateCaep-RecoRcp-lastMCK_mb-poisson-0.5_4032_02119114525 -->
/pnfs/sam/lto/copy1/monte_carlo/phase10/mcc99/reco/all/reco_mcp10_p10.15.01_nikhef_pythia_calibv00+03+27
EtaGt-4.2+TM-174.3+PtGt5.0+KinMGt-60.0+EtaLt4.2+KinMLt-130.0-PlateCaep-RecoRcp-lastMCK_mb-poi
0.5_4032_02119114525

D31DLTO.mover alive : SEEK d0enmvr17a 2002-May-10 14:34:08

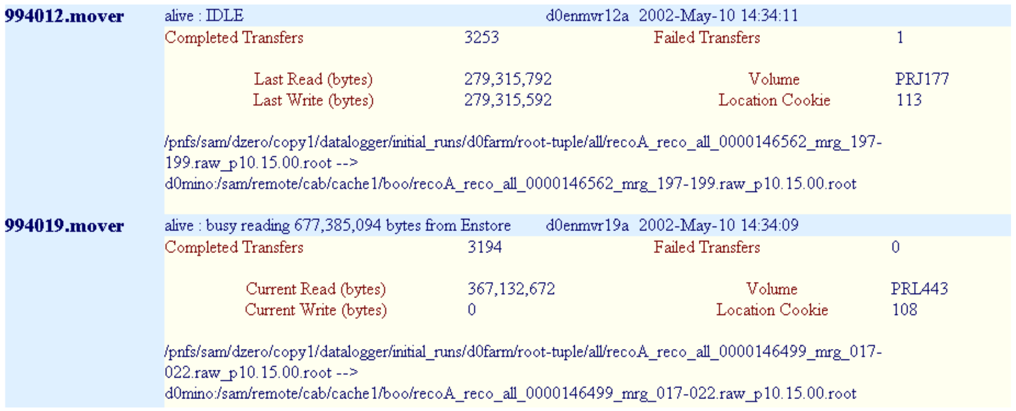
Completed Transfers	1587	Failed Transfers

D31ELTO.mover alive : busy writing 1,536,695,557 bytes to Enstore d0enmvr10a 2002-May-10 14:34:05

Completed Transfers	1405	Failed Transfers	0
Current Read (bytes)	701,759,311	Volume	PRK217L1
Current Write (bytes)	655,884,288	EOD Cookie	151

d0bbin:/d0/stripe7/samtest/14573/store_in_progress/reco_all_0000153407_028.raw_p10.15.01_000 -->
/pnfs/sam/lto/copy1/datalogger/initial_runs/d0farm/reco/all/reco_all_0000153407_028.raw_p10.15.01_000

This image shows movers that are idle (awaiting a job), and busy reading a tape:



994012.mover alive : IDLE d0enmvr12a 2002-May-10 14:34:11

Completed Transfers	3253	Failed Transfers	1
Last Read (bytes)	279,315,792	Volume	PRJ177
Last Write (bytes)	279,315,592	Location Cookie	113

/pnfs/sam/dzero/copy1/datalogger/initial_runs/d0farm/root-tuple/all/recoA_reco_all_0000146562_mrg_197-199.raw_p10.15.00.root -->
d0mino:/sam/remote/cab/cache1/boo/recoA_reco_all_0000146562_mrg_197-199.raw_p10.15.00.root

994019.mover alive : busy reading 677,385,094 bytes from Enstore d0enmvr19a 2002-May-10 14:34:09

Completed Transfers	3194	Failed Transfers	0
Current Read (bytes)	367,132,672	Volume	PRL443
Current Write (bytes)	0	Location Cookie	108

/pnfs/sam/dzero/copy1/datalogger/initial_runs/d0farm/root-tuple/all/recoA_reco_all_0000146499_mrg_017-022.raw_p10.15.00.root -->
d0mino:/sam/remote/cab/cache1/boo/recoA_reco_all_0000146499_mrg_017-022.raw_p10.15.00.root

Understanding the Number of Bytes Read/Written

For a READ job,

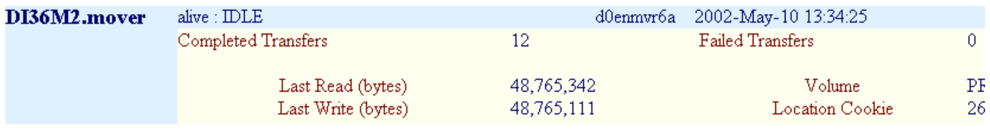
"Last/Current Read (bytes)"	means "bytes read from tape"
"Last/Current Write (bytes)"	means "bytes written to user's file"

whereas for a WRITE job,

"Last/Current Read (bytes)"	means "bytes read from user's file"
"Last/Current Write (bytes)"	means "bytes written to tape"

For jobs in progress, the number of "Current Read (bytes)" is by necessity higher than "Current Write (bytes)".

For finished jobs (e.g., of status IDLE or busy dismounting volume), you can compare "Last Read (bytes)" to "Last Write (bytes)" to tell if a job was a READ or WRITE. The file size is always bigger on tape than on the user's disk because the file family wrapper is on the tape copy only. So for example, on a READ job, Enstore reads a larger file from tape and writes a smaller one to disk, and thus the "Last Read (bytes)" value is larger than "Last Write (bytes)" (as shown in image below). The converse is true for a WRITE job.



DI36M2.mover	alive : IDLE		d0enmvr6a	2002-May-10 13:34:25	
	Completed Transfers	12	Failed Transfers		0
	Last Read (bytes)	48,765,342	Volume		PF
	Last Write (bytes)	48,765,111	Location Cookie		26

9.10 Encp History

What?	This page lists the last several **encp** transfers that have completed, either successfully or with an error.
Why?	Use this page to review recent **encp** transfers.
How?	To arrive at the **Encp History** page, click "Encp History" on the top page or the **ENCP** button at the top of any page.

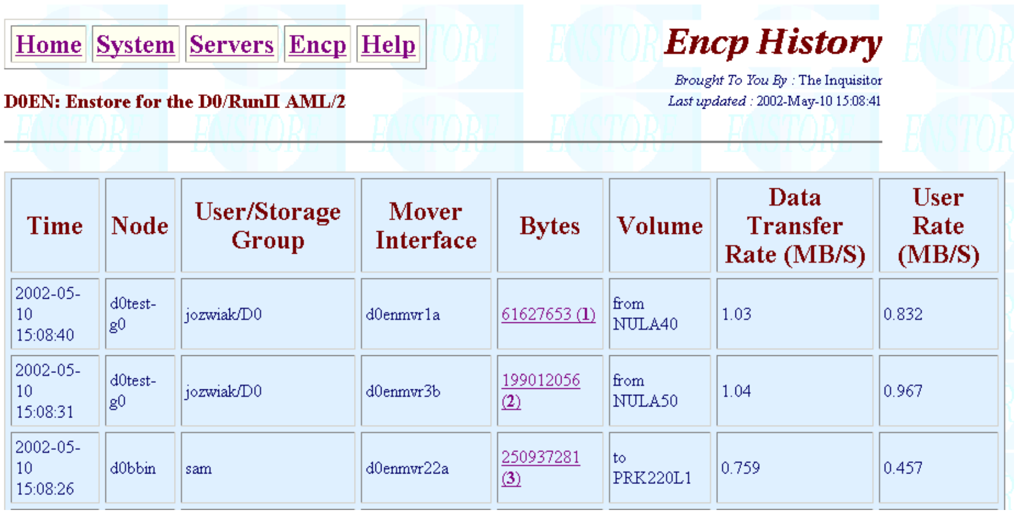
Page Description

On the **Encp History** page:

Successful transfers show	time that transfer completed, node, username and storage group, mover interface (the TCP/IP interface used on the mover node), bytes transferred, volume ID, and data transfer rate and user rate (both in Mb/s)

Unsuccessful transfers show	time of attempted transfer, node, username, storage group and error summary. Each error summary contains a link to a more detailed error message.

The top portion of the page is a table listing details of each recent transfer:



Home | System | Servers | Encp | Help

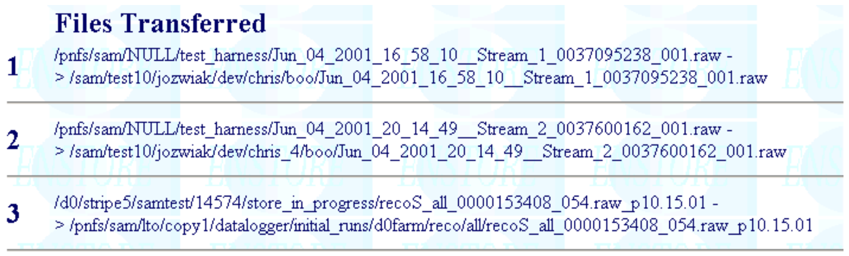
Encp History

Brought To You By : The Inquisitor
Last updated : 2002-May-10 15:08:41

D0EN: Enstore for the D0/RunII AML/2

Time	Node	User/Storage Group	Mover Interface	Bytes	Volume	Data Transfer Rate (MB/S)	User Rate (MB/S)
2002-05-10 15:08:40	d0test-g0	jozwiak/D0	d0enmvr1a	61627653 (1)	from NULA40	1.03	0.832
2002-05-10 15:08:31	d0test-g0	jozwiak/D0	d0enmvr3b	199012056 (2)	from NULA50	1.04	0.967
2002-05-10 15:08:26	d0bbin	sam	d0enmvr22a	250937281 (3)	to PRK220L1	0.759	0.457

The value under *Bytes* provides a link to the *Files Transferred* area of the page which gives you the originating and destination file names:



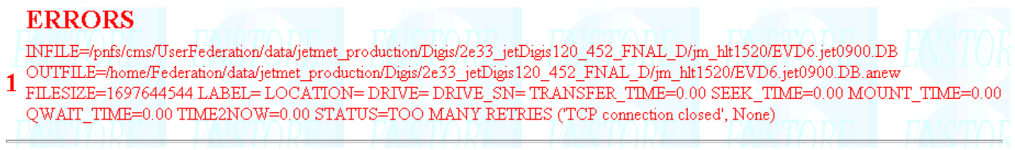
Files Transferred

1 /pnfs/sam/NULL/test_harness/Jun_04_2001_16_58_10__Stream_1_0037095238_001.raw -
> /sam/test10/jozwiak/dev/chris/boo/Jun_04_2001_16_58_10__Stream_1_0037095238_001.raw

2 /pnfs/sam/NULL/test_harness/Jun_04_2001_20_14_49__Stream_2_0037600162_001.raw -
> /sam/test10/jozwiak/dev/chris_4/boo/Jun_04_2001_20_14_49__Stream_2_0037600162_001.raw

3 /d0/stripe5/samtest/14574/store_in_progress/recoS_all_0000153408_054.raw_p10.15.01 -
> /pnfs/sam/lto/copy1/datalogger/initial_runs/d0farm/reco/all/recoS_all_0000153408_054.raw_p10.15.01

At the very bottom of the page, you can find the errors in red, if any:



ERRORS

```
1 INFILE=/pnfs/cms/UserFederation/data/jetmet_production/Digis/2e33_jetDigis120_452_FNAL_D/jm_hlt1520/EVD6.jet0900.DB
  OUTFILE=/home/Federation/data/jetmet_production/Digis/2e33_jetDigis120_452_FNAL_D/jm_hlt1520/EVD6.jet0900.DB.anew
  FILESIZE=1697644544 LABEL= LOCATION= DRIVE= DRIVE_SN= TRANSFER_TIME=0.00 SEEK_TIME=0.00 MOUNT_TIME=0.00
  QWAIT_TIME=0.00 TIME2NOW=0.00 STATUS=TOO MANY RETRIES ('TCP connection closed', None)
```

9.11 Configuration

What? The **Enstore Configuration** page shows the Enstore system's current configuration.

Why? This page is for administrators. But if you want configuration information on any component in the Enstore system, you can look here. For example:

- If you see a server listed as unmonitored (in grey) on the **Enstore Server Status** page, you can verify its status here (if the element `inq_ignore` appears, the server is unmonitored).
- If you want to check the log files for activity related to a particular mover, look here for the `logname` value associated with the mover, then search the log files for that string.

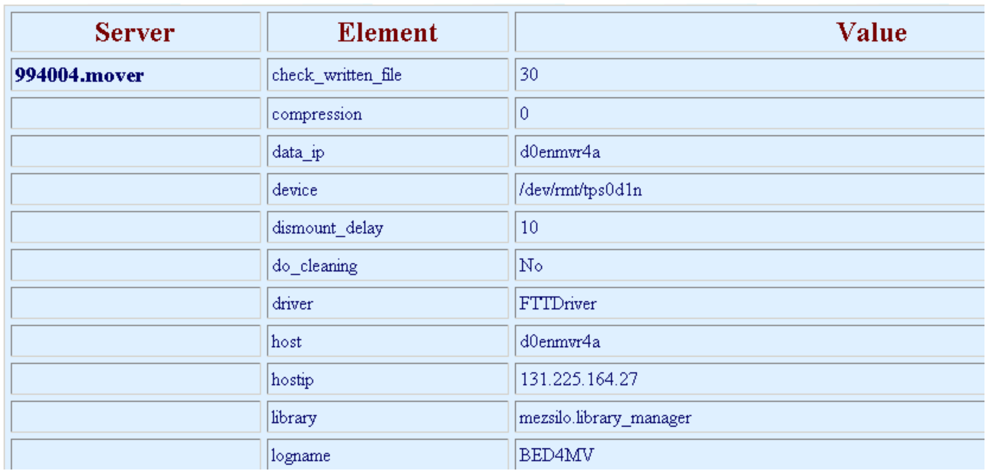
How? To arrive at the **Enstore Configuration** page, click "Configuration" on the top page.

Page Description

The page is divided into two sections:

- The first section provides a quick link to each of the servers listed in the table in the second section.
- The second section, is a (potentially quite long) table containing detailed configuration information for all of the Enstore servers. For each server, there is a table row for each element that appears in the server's configuration. The information displayed includes the element name and its current value. No interpretation of the values is done, so for instance if the value is a python dictionary, then it is presented here as such. The server names, and under them the element names, are organized alphabetically.

This image shows the top of the table in the second section on the **Enstore Configuration** page, including the (truncated) entry for one of the system's movers:



Server	Element	Value
994004.mover	check_written_file	30
	compression	0
	data_ip	d0enmvr4a
	device	/dev/rmt/tps0d1n
	dismount_delay	10
	do_cleaning	No
	driver	FTTDriver
	host	d0enmvr4a
	hostip	131.225.164.27
	library	mezsilo.library_manager
	logname	BED4MV

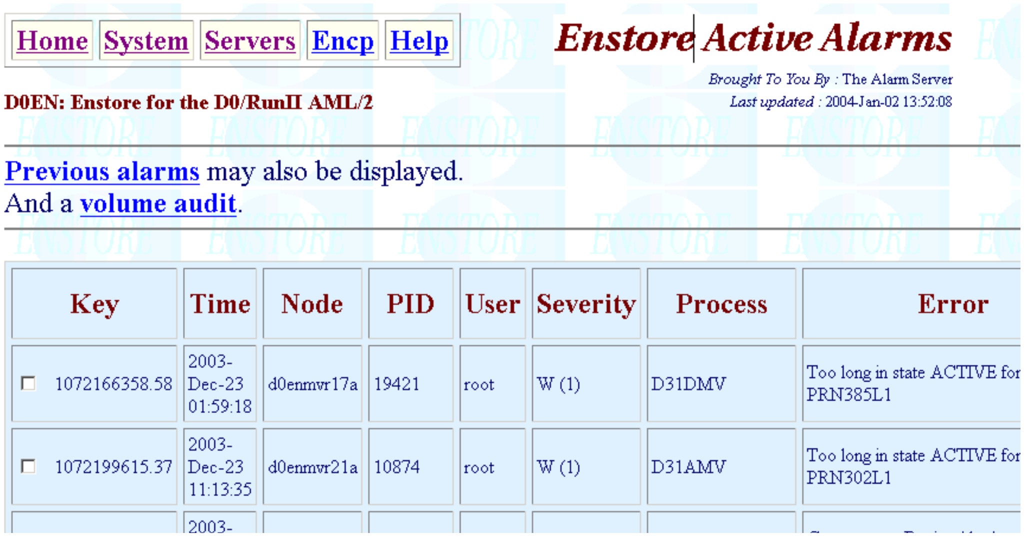
9.12 Enstore Active Alarms

What? The **Enstore Active Alarms** page lists the alarms that have been raised but not yet resolved.

Why? This page is for administrators, but as a user, you can always look here for information when there is a problem with Enstore. In particular, if a volume is set to NOACCESS, you can look here to find out which mover was involved.

How? To arrive at the **Enstore Active Alarms** page, click "Alarms" on the top page.

The page is quite wide; we show first the left side, then the right.



Home | System | Servers | Encp | Help

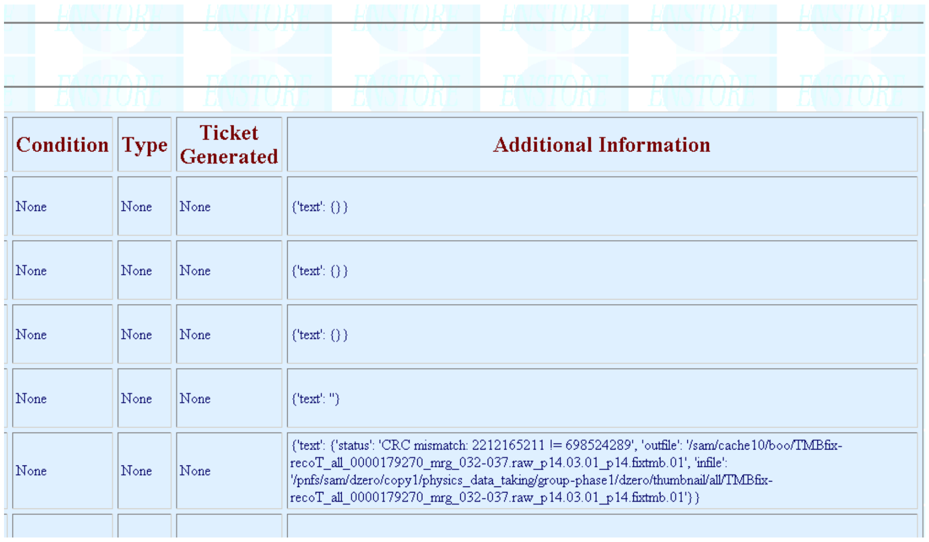
Enstore Active Alarms

Brought To You By : The Alarm Server
Last updated : 2004-Jan-02 13:52:08

D0EN: Enstore for the D0/RunII AML/2

Previous alarms may also be displayed.
And a volume audit.

Key	Time	Node	PID	User	Severity	Process	Error
☐ 1072166358.58	2003-Dec-23 01:59:18	d0enmvr17a	19421	root	W (1)	D31DMV	Too long in state ACTIVE for PRN385L1
☐ 1072199615.37	2003-Dec-23 11:13:35	d0enmvr21a	10874	root	W (1)	D31AMV	Too long in state ACTIVE for PRN302L1
	2003-						



Condition	Type	Ticket Generated	Additional Information
None	None	None	{'text': {} }
None	None	None	{'text': {} }
None	None	None	{'text': {} }
None	None	None	{'text': ''}
None	None	None	{'text': {'status': 'CRC mismatch: 2212165211 != 698524289', 'outfile': '/sam/cache10/boo/TMBfix-recoT_all_0000179270_mrg_032-037.raw_p14.03.01_p14.fixtmb.01', 'infile': '/pnfs/sam/dzero/copy1/physics_data_taking/group-phase1/dzero/thumbnail/all/TMBfix-recoT_all_0000179270_mrg_032-037.raw_p14.03.01_p14.fixtmb.01'} }

9.13 Enstore Log Files

What? The **Enstore Log Files** page provides links to Enstore system-specific user log files and to standard Enstore daily log files. You can search log files or retrieve entire log files.

Why?	This page is for administrators. You can use the log files to retrace Enstore activity, to understand Enstore problems or behavior, and so on.
How?	To arrive at the **Enstore Log Files** page, click "Log Files" on the top page.

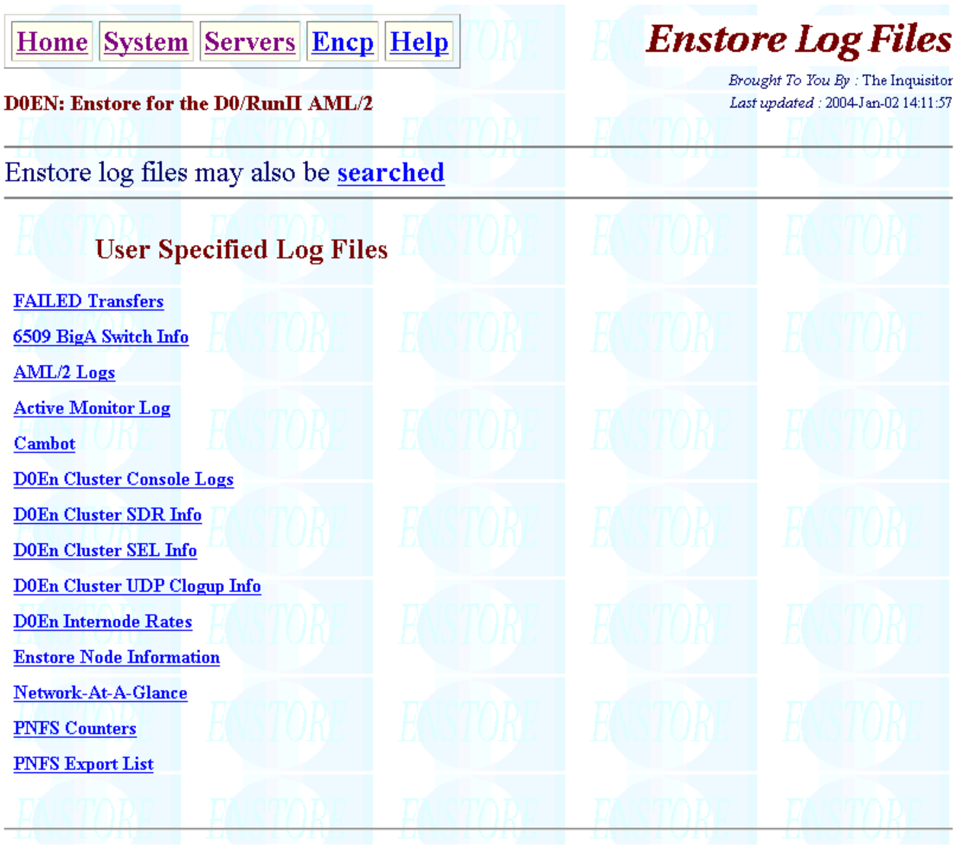
There are three sections to the **Enstore Log Files** page.

Link to Search Page

First there is a link to a search page; look for "Enstore log files may also be *searched*". Use the **HELP** button for information on constructing your search string. (Shown in the image below.)

User Specified Log Files

The next part is entitled *User Specified Log Files*. It lists miscellaneous log files configured and maintained for your Enstore system.


Home System Servers Encp Help

Enstore Log Files

Brought To You By : The Inquisitor
Last updated : 2004-Jan-02 14:11:57

D0EN: Enstore for the D0/RunII AML/2

Enstore log files may also be searched

User Specified Log Files

FAILED Transfers
6509 BigA Switch Info
AML/2 Logs
Active Monitor Log
Cambot
D0En Cluster Console Logs
D0En Cluster SDR Info
D0En Cluster SEL Info
D0En Cluster UDP Clogup Info
D0En Internode Rates
Enstore Node Information
Network-At-A-Glance
PNFS Counters
PNFS Export List

Any given Enstore installation may contain some or all of these log files:

FAILED Transfers	Lists all encp jobs that failed; lists by volume and by mover

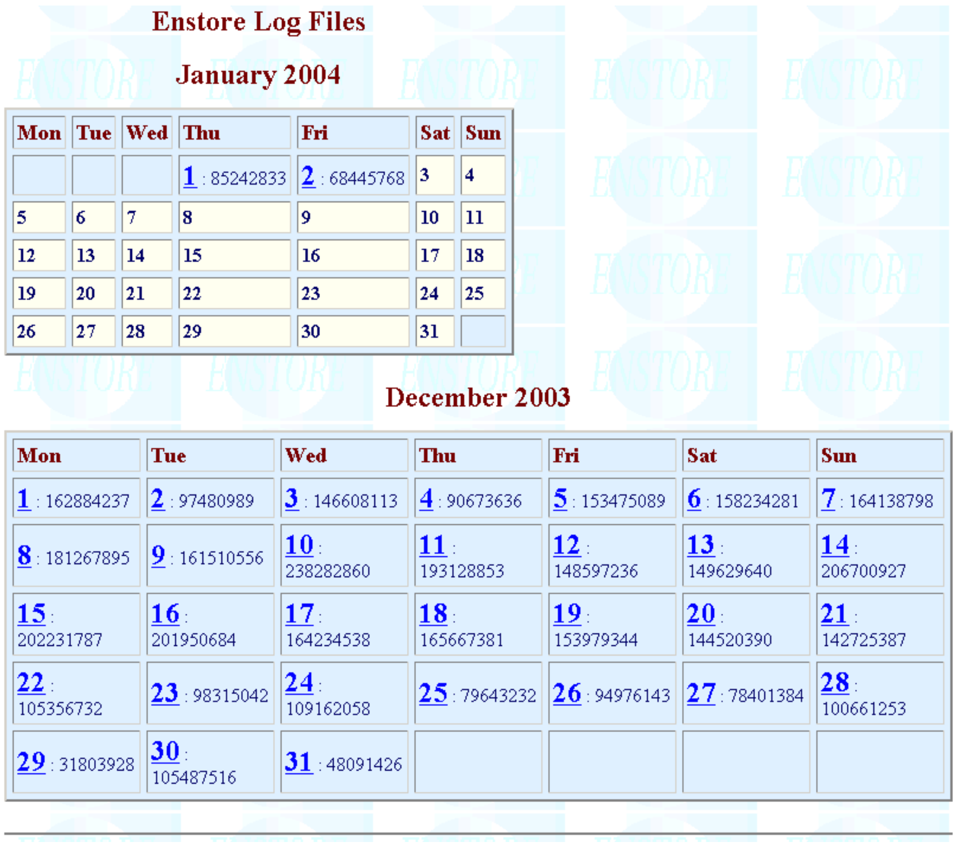
Recent (robot) log messages	Displays all the messages from the robot (for the most recent few days)
Active Monitor Log	Displays the data transfer rate between the base node and all other nodes in the same Enstore system, including movers
Cambot (D0)	Displays a live image photographed by a camera mounted inside the D0 ADIC robot
Enstore Node Information	Displays information on all nodes belonging to this Enstore system.
Network-At-A-Glance	Displays network interface status of all nodes relative to base node; uses colored icons for easy identification of problems
PNFS Counters	Displays how much space is used in each PNFS directory
PNFS Export List	Lists all the existing PNFS areas for the Enstore system (when PNFS is mounted, these are the areas that get NFS-mounted)
Cluster UDP Clogup Info	Displays UDP activity in the Enstore system
Cluster Console Logs	logs from consoles
Cluster SDR Info	Displays hardware information, e.g., temperatures, fan speeds, voltages, and so on, for all nodes (servers and movers)
Cluster SEL Info	Displays System Event Log for all nodes in system
Internode Rates or Network Rate Test	Displays data transfer rates between each node and the base node[1]
6509 BigA Switch Info (D0)	Displays information relating to switch's status

Enstore Log Files

The bottom portion of the page is called *Enstore Log Files*. It displays a calendar of the current and previous months from which you can click the date of the log file to view (the image below was captured on January 2, 2004, the last date that shows an entry). The size of the log file is given.

1. The Active Monitor Log uses a different base node (<system>srv2) from the Internode Rates or Network Rate (<system>srv3).

Requesting a day’s log file is memory intensive and very slow due to the large size of the log file.



Enstore Log Files

January 2004

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1 : 85242833	2 : 68445768	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

December 2003

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1 : 162884237	2 : 97480989	3 : 146608113	4 : 90673636	5 : 153475089	6 : 158234281	7 : 164138798
8 : 181267895	9 : 161510556	10 : 238282860	11 : 193128853	12 : 148597236	13 : 149629640	14 : 206700927
15 : 202231787	16 : 201950684	17 : 164234538	18 : 165667381	19 : 153979344	20 : 144520390	21 : 142725387
22 : 105356732	23 : 98315042	24 : 109162058	25 : 79643232	26 : 94976143	27 : 78401384	28 : 100661253
29 : 31803928	30 : 105487516	31 : 48091426				

9.14 Quota and Usage

What? The **Quota and Usage** page provides information on your Enstore volume usage, organized by library and by storage group. The page is not real-time, it displays a recent snapshot.

Why? Administrators and users can look here to see a variety of details about your Enstore system’s resource and quota management.

How? To arrive at the **Quota and Usage** page, click “Quota and Usage” on the top page.

This page displays the following fields:

Library	library manager

Storage Group	storage group
Req. Alloc.	requested volume (e.g., tape) allocation
Auth. Alloc.	authorized volume allocation
Quota	total space allowed in robot
Allocated	total number of volumes currently allocated in the robot
Blank Vols	of the allocated volumes, the number that are blank
Used Vols	of the allocated volumes, the number that are written
Deleted Vols	of the allocated volumes, the number that have been deleted
Space Used	total space used on all allocated volumes
Active Files	total number of active (non-deleted) files on all allocated volumes
Deleted Files	total number of deleted files on all allocated volumes
Unknown Files	

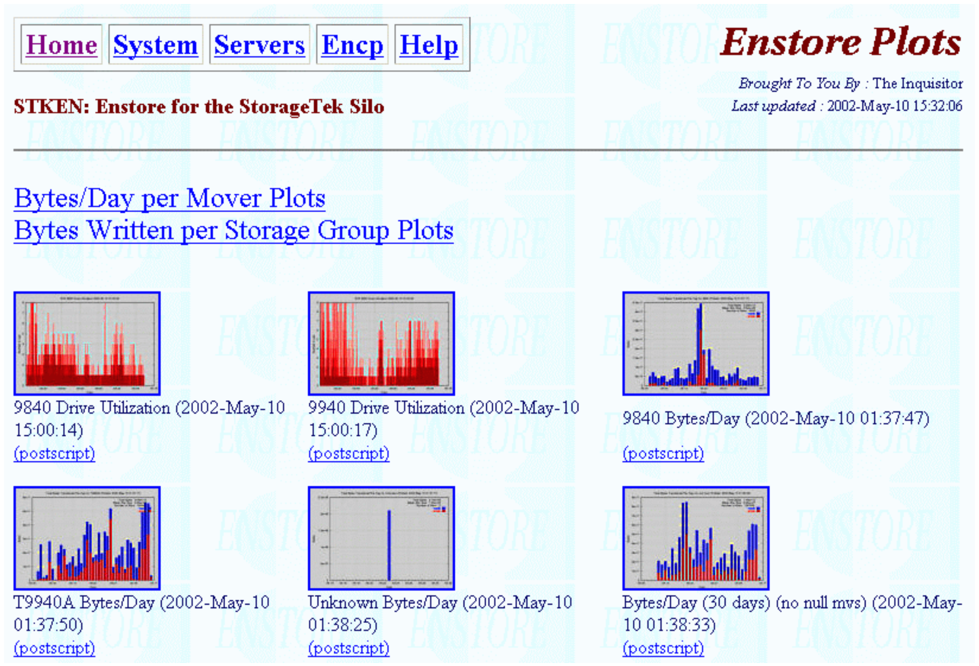
9.15 Enstore Plots

What?	The **Enstore Plots** page provides information on Enstore performance in a visual format. These are not real-time, they are snapshots.
Why?	You can look here to see a variety of details about your Enstore system's recent performance.
How?	To arrive at the **Enstore Plots** page, click "Plots" on the top page.

The **Enstore Plots** page, provides information on some statistics that Enstore gathers. These statistics are gathered from the log files produced by Enstore. Several plots are available:

- Drive Utilization
- Bytes/Day Plot
- Bytes/Day per Mover Plot
- Mount Latency Plot
- Mounts/Day per Drive Type

- Storage group activity (STKEN only)
- Total bytes/Day
- Total bytes/Day combining all three systems (D0EN, CDFEN, STKEN)
- Total bytes written/Day
- Cumulative Mounts Plot
- Transfer Activity (log) Plot
- Transfer Activity Plot
- Mounts/Day Plot
- Null Terabytes/Day (Instantaneous Rate Plot)
- Real Terabytes/Day (Instantaneous Rate Plot)


Home
System
Servers
Encp
Help
Enstore Plots
Brought To You By : The Inquisitor
Last updated : 2002-May-10 15:32:06
STKEN: Enstore for the StorageTek Silo
Bytes/Day per Mover Plots
Bytes Written per Storage Group Plots
9840 Drive Utilization (2002-May-10 15:00:14)
(postscript)
9940 Drive Utilization (2002-May-10 15:00:17)
(postscript)
9840 Bytes/Day (2002-May-10 01:37:47)
(postscript)
T9940A Bytes/Day (2002-May-10 01:37:50)
(postscript)
Unknown Bytes/Day (2002-May-10 01:38:25)
(postscript)
Bytes/Day (30 days) (no null mvs) (2002-May-10 01:38:33)
(postscript)

All the plots are described in the online help page. Each plot is available for viewing three ways:

- a small version of the plot (postage stamp) displayed directly on the page
- a full size version of the plot; click on the postage stamp to display
- a postscript copy of the plot; click on the (postscript) link to display

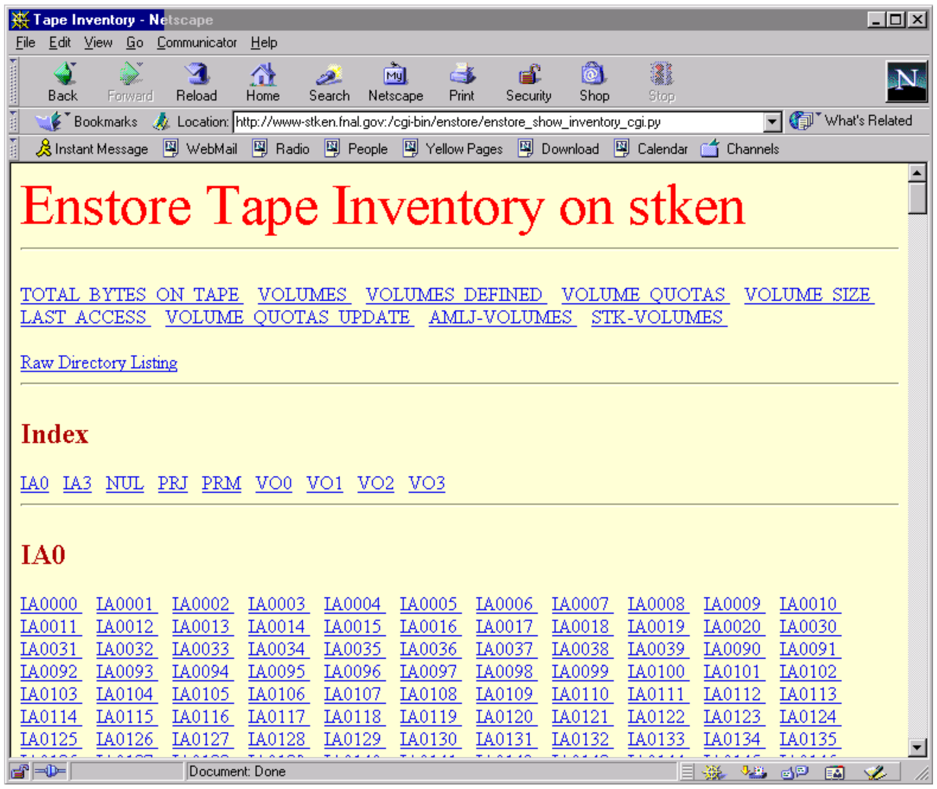
9.16 NGOP Monitoring

NGOP stands for “Next Generation OPerations”. This link requires a password to get in, and is typically reserved for administrators. It displays information relating to the Enstore node, e.g., daemons, the operating system, hardware, cron jobs, enstore nodes, and so on.

9.17 Tape Inventory Page (Dynamic HTML)

What?	For each volume declared to your Enstore system you can dynamically (re)create a page that presents its volume inventory information in HTML format.
Why?	Use this page to find out details of the storage of your file(s) on a volume, to see how full a tape is, or to check the inhibits.
How?	To arrive at this page start at the top page, scroll down to the *Information* table, and click “Tape Inventory”. This brings you to a list of all declared volumes in your Enstore system. Click on the volume of interest.

This page is dynamic and uses a lot of server time; please minimize the number of times you regenerate this page.


Tape Inventory - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://www-stken.fnal.gov:/cgi-bin/enstore/enstore_show_inventory_cgi.py
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Document: Done

Enstore Tape Inventory on stken

TOTAL BYTES ON TAPE VOLUMES VOLUMES DEFINED VOLUME QUOTAS VOLUME SIZE
LAST ACCESS VOLUME QUOTAS UPDATE AMLJ-VOLUMES STK-VOLUMES

Raw Directory Listing

Index

IA0 IA3 NUL PRJ PRM VO0 VO1 VO2 VO3

IA0

IA0000	IA0001	IA0002	IA0003	IA0004	IA0005	IA0006	IA0007	IA0008	IA0009	IA0010
IA0011	IA0012	IA0013	IA0014	IA0015	IA0016	IA0017	IA0018	IA0019	IA0020	IA0030
IA0031	IA0032	IA0033	IA0034	IA0035	IA0036	IA0037	IA0038	IA0039	IA0090	IA0091
IA0092	IA0093	IA0094	IA0095	IA0096	IA0097	IA0098	IA0099	IA0100	IA0101	IA0102
IA0103	IA0104	IA0105	IA0106	IA0107	IA0108	IA0109	IA0110	IA0111	IA0112	IA0113
IA0114	IA0115	IA0116	IA0117	IA0118	IA0119	IA0120	IA0121	IA0122	IA0123	IA0124
IA0125	IA0126	IA0127	IA0128	IA0129	IA0130	IA0131	IA0132	IA0133	IA0134	IA0135

Find the volume of interest and click it to get a file listing. The format is essentially identical to that shown in section 9.8 *Tape Inventory Page (Text)*, but in addition under the heading, you get a list of volume parameters:

```
{'blocksize': 131072,
 'capacity_bytes': 20401094656L,
 'declared': 998086428.80971301,
 'eod_cookie': '0000_000000000_0000446',
 'external_label': 'VO0094',
 'first_access': 998268618.43976796,
 'last_access': 1013139175.787874,
 'library': 'eagle',
 'media_type': '9840',
 'non_del_files': 445,
 'remaining_bytes': 2269696L,
 'sum_rd_access': 35,
 'sum_rd_err': 0,
 'sum_wr_access': 445,
 'sum_wr_err': 0,
 'system_inhibit': ['none', 'full'],
 'user_inhibit': ['none', 'none'],
 'volume_family': 'theory.theory-canopy-eichten.cpio_odc',
 'wrapper': 'cpio_odc'}
```

Chapter 10: Job Priority and Queue Management

Users submit read and write jobs to **encp** (often through an interface, e.g., dCache). **Encp** sends a request for each job to an Enstore library manager for processing. The library manager receives these requests, stores them in a queue, assigns a priority to each one, and passes them to a mover for actual data transfer. A request's priority determines when it will get processed relative to others in the queue, and its priority may change as circumstances change while it waits in the queue. There are four items that factor into determining priority: category, numerical value within the category, "Fair Share", and ownership of resources by a group/experiment.

10.1 Job Priority Categories

There are two categories of job priority: normal and DAQ/Admin. The default priority is "normal". DAQ/Admin priority, as its name implies, is reserved for high-priority jobs.

DAQ/Admin priority is granted only to job requests that satisfy certain preconfigured conditions. Conditions, if set, filter on the request's originating username, group, node, and so on. For example, conditions could be set to grant DAQ/Admin priority to all jobs submitted by user *joe* from node *mynode1*, by users *george* or *kim* from any node, and by any user from node *myspecialnode*; all other jobs get normal priority.

☞ Setting conditions for DAQ/Admin priority is an administrative function.

Requests that get submitted and receive DAQ/Admin priority get moved to the head of the LM's request queue. If there is more than one at a time, the other priority-related factors determine the order in which these requests get processed. Any normal priority transfer that is in progress is allowed to complete, but the system does not then process other normal priority requests in the queue that are waiting for the same volume. At the completion of the current transfer, the tape is replaced in the drive if necessary, then the (first) DAQ/Admin request gets processed.

10.2 Numerical Priority Values

Within its priority category, each request is also assigned an initial numeric priority value. The numeric priority is set by default according to preconfigured conditions. It can be overridden on the command line using the **encp** options described in section 5.5 *Encp Command Options*, although we strongly recommend against doing this. The numeric priority may change while the request waits in the queue depending on (a) the **encp** options used, (b) the elapsed time in the queue.

10.3 Fair Share Resource Allotment

Enstore queue management has an algorithm called “Fair Share”, that helps to keep any one storage group (experiment or group) from monopolizing tape drives.

When a mover which does not have a mounted tape asks the Library Manager for the next request, the Fair Share determines which storage groups currently have requests in progress (at a mover) and which ones don’t, then gives preference to requests associated with those that don’t. This helps ensure that there is a fair distribution of resources available to all groups currently using the system.

10.4 Resource Ownership

When an experiment or group purchases one or more tape drives, the drives go into the pool of Enstore resources, accessible by all users, with the recognition that the purchasing group has preferential access to this number of drives. The Fair Share configuration (section 10.3) gets modified to guarantee this access. The priorities of other requests in the queue may be perturbed when Enstore needs to free up one or more tape drives for jobs submitted by the purchasing group.

In contrast to DAQ/Admin requests, these (normal priority) requests, identified by their storage group, must wait until a tape has dismounted in the normal way before being processed. A tape normally dismounts after all requests in the queue requiring that tape have completed.

Appendix A. Network Control

This appendix discusses the control of ethernets on client systems when the client interacts directly with Enstore via encp, as opposed to interacting with the dCache as a front-end.

A.1 Default Routing for Encp

By default, encp uses the DNS name obtained by the **`hostname`** command for control messages. Encp uses whatever routing the client system or network administrator sets between the client system and the Enstore system for data transfers.

Typically, the default configuration suffices for machines having a single network interface, or having a single network interface dedicated to data movement.

A.2 Routing via the enstore.conf File

For large client machines with multiple network connections (interfaces), each network interface is attached to a different (virtual) router. A one-to-one mapping is made between the IP address of each interface and the IP of the router it is attached to. The Computing Division's networking group must perform the configuration for this.

Administrators can create an `enstore.conf` file to configure the routing to allow for multiple network interface cards dedicated to Enstore, and/or to allow for a different IP address to be used for the encp control socket. The default location for the file is `/etc/enstore.conf`. The location of the file can be overridden with the environment variable ENSTORE_CONF.

The file format supports comments, a host ip line, and zero or more interface lines (all these line types are optional).

Comment Lines

Comment lines begin with a "pound" sign, "#", e.g.,:

```
# this is s comment line
```

Hostip Line

The `hostip` line gives the host IP address used to override the DNS name that encp uses to **`bind`** with. This is used when doing a (passive) open on a socket used to listen for a call back from the mover. One and only one hostname line is required per `enstore.conf` file. For example:

```
hostip=131.225.42.42
```

Interface Lines

Lines starting with `interface` are used in this file to specify more than one network interface for data transfers. An `enstore.conf` file would typically contain either zero or at least two of these lines, since a single interface can be controlled more conventionally with static routes.

This functionality is known to work on IRIX machines. It has not been tested on SunOS or OSF1, although it is expected to work; the Linux kernel doesn't support it.

The underlying implementation mechanisms for multiple interfaces are portable, and the scheme can be extended on demand. When more than two interfaces are used, it is necessary to have each interface on its own subnet. (The system or network administrator needs to configure the subnets.) The enroute2 executable (part of the encp product) must be installed and have setuid root on the machine in order to enable this feature.

In order for the interface lines in the `/etc/enstore.conf` to be used; an executable named `enroute2` with the setuid bit turned on needs to be in **encp**'s path. This executable is included with the **encp** product from **UPS/UPD**, but the setuid bit is not set by default. The search path for this executable with the setuid bit set is:

1) $ENROUTE2

2) $ENCP_DIR/enroute2

3) $ENSTORE_DIR/enroute2

4) /usr/local/bin/enroute2

5) /etc/enroute2

An `interface` line must specify four keywords (with an optional fifth for IRIX):

- `interface` specifies the network device.
- `weight` specifies the relative capacity of each interface. For example, if 1 Gb/s and 100 Mb/s interfaces are used, they might be assigned weights of 30 and 10, respectively[1].
- `ip` specifies the ip address corresponding to the device given by `interface`.

- `gw` specifies the ip address corresponding to the gateway to the Enstore movers for the device given by `interface`. (Get this information from the networking admins.)
- `cpu` The `cpu` keyword is used on IRIX systems only. Its use is desirable for minimizing the amount of CPU used per transfer, though it is technically not required. The performance enhancements will take effect if two conditions are met: If the CPU used by encp has hardware affinity with the slot holding the network card, and if the same CPU performs interrupt service for the network card.

For example, a file may contain two `interface` lines as follows:

```
interface=eg2 weight=30 cpu=2 gw=131.225.32.32 ip=131.225.32.31
interface=eg3 weight=10 cpu=3 gw=131.225.32.36 ip=131.225.32.35
```

1. Despite the 1Gb name, these cards typically get 30 MB/s. A 100Mb/s network card typically gets a 10/MB/s rate. Hence, the weights 30 and 10.

Appendix B. Changing PNFS Tags

B.1 Caveat

Tags (i.e., tag values) can be changed if the standard UNIX permissions on them allow for it. However, thought and planning should go into structuring the storage of an experiment's data, and users should not change any tags without consulting the people in their experiment responsible for this task. The storage group tag cannot be changed by users.

B.2 Permissions and Ownership

☞ This is typically done by your experiment's Enstore liaison. Most users do not have permissions to change permissions and ownership.

The permissions shown in the output of the **`enstore pnfs --tags`** command (see section 3.2.2 *How to View Tags*) indicate whether you can change the value of the tag or not. To change a tag, you need to use the **`enstore pnfs`** command with one of the options **`--tagchown`** or **`--tagchmod`** to change ownership or permissions, respectively (see section 8.4 *enstore pnfs*).

For example, to add write permission for "other" to the permissions for the file family tag, you'd enter (include the quotes):

```
% enstore pnfs --tagchmod o+w file_family
```

or you can use the absolute form for the mode, e.g.,

```
% enstore pnfs --tagchmod [0]646 file_family
```

To change the ownership, run a command like either of the following, using userid or userid.groupid, or any associated combination:

```
% enstore pnfs --tagchown xyz file_family
```

```
% enstore pnfs --tagchown xyz.g023 file_family
```

B.3 How to Set a Tag

To set one or more tags on a directory, `cd` to that directory and run the **`enstore pnfs`** command with the option for the tag you want to reset. It is ok to change tags on a directory into which files have already been written; Enstore will still be able to find the files.

These operations are for Enstore admins or designated gurus only; the commands succeed only if permissions allow.

--file-family <FILE_FAMILY_NAME>

If permissions allow, this will reset the file-family name of the current pnfs directory to the specified value. Example:

```
$ enstore pnfs --file-family newfamilyname
```

--file-family-width <FILE_FAMILY_WIDTH>

If permissions allow, this will reset the file-family width of the current pnfs directory to the specified value. Example:

```
$ enstore pnfs --file-family-width 2
```

--file-family-wrapper <FILE_FAMILY_WRAPPER>

If permissions allow, this will reset the file-family wrapper of the current directory to the specified value. Example:

```
$ enstore pnfs --file-family-wrapper cpio_odc
```

--library <LIBRARY>

Provides a name to reset the library tag. Example:

```
$ enstore pnfs --library mylib
    [Errno 13] Permission denied: '/pnfs/test/xyz/srmtest/test_1_1/.(tag)(library)'
```

Run the **`enstore pnfs --tags`** command to see the changes you make; see section 8.4 *enstore pnfs.*

Appendix C. FTP: Problems in Grid Environment

C.1 Problems with Partial Files

Typical FTP clients present problems when transferring files in a grid environment. The typical FTP clients do not transfer files such that a server can tell the difference between an end-of-file and a partial transfer.

Problem 1: Since a server cannot distinguish between the end-of-file and a dropped/aborted transfer, partial files may end up in the underlying mass storage system.

Problem 2: There is a server acknowledge back to the client at the end of a transfer, but it is not a handshake. Therefore, the server doesn’t actually know whether the client is there or not.

The **`size`** command gets around these problems, but is usually not required for transferring files. Imposing use of this command would be incompatible with numerous ftp clients. Similarly, the ftp block mode contains enough information to determine if a complete file has been sent or if a transfer has been aborted. This mode is defined in the standard, but is not in widespread use.

For either solution (size or block), normal ftp installations would be expected (although not required) to keep the partial files so that the user can complete the transfer. For a grid connected to a storage system, these partial files are an added nuisance, although not a severe one (the system could just throw them away).

C.2 ACL Timeouts

Many networks have ACL limitations imposed on them. A common ACL is the "reflexive" ACL, for which a remote client allows a connection only if the local side is the initiator. These reflexive ACLs have timeouts usually in the minutes, and almost always less than 30 minutes. We must walk the line between lengthening the time window enough to decrease the number of timeouts but not so much as to increase the security risks.

Transfers that take a long time to complete (slow link, big files) are the common victims. These transfers involve plenty of activity on the FTP data socket, but no traffic on the FTP control socket. The ACL on the FTP control times out and the server cannot send the client its acknowledgement. Moreover, the client can’t send more commands without stopping and restarting (because the control port connection is broken).

C.3 Third Party Passive Transfers Impossible

Third party passive transfers are not possible. Typically, passive transfers are used in the cases of firewalls and reflexive ACLs. The FTP passive protocol requires that control and data ports be negotiated before the filename is sent. Since data could be in many places (e.g., in different storage pools), the server has no way of negotiating ports for an unknown, and more than likely different, computer. The only way around this problem is to introduce an “adaptor tunnel”, whereby the file is first transferred from the storage pool to a known computer and then to the client’s computer. This gets around the port negotiation, but introduces a scalability problem. There is a bottleneck introduced in the adaptor.

Enstore Glossary

accounting server
: This server maintains statistical information on a running system. It is Enstore's interface to a database of transfer-related data.

active file
: Any file in Enstore that is not deleted.

alarm server (AS)
: The Alarm Server maintains a record of alarms raised by other servers, and creates a report that's available online.

bfid
: Bit file id; an Enstore-assigned, unique identifier for a data file.

cern wrapper
: A file family wrapper that accommodates data files up to (2^64 – 1) bytes. It matches an extension to the ANSI standard, as proposed by CERN, and allows data files written at Fermilab to be readable by CERN, and vice-versa. See **file family wrapper**.

configuration server (CS)
: The Configuration Server maintains and distributes the information about Enstore system configuration, such as the location and parameters of each Enstore component and/or server.

cpio_odc wrapper
: A file family wrapper which allows the file to be dumpable via cpio. This wrapper has a file length limit of (8G – 1) bytes. See **file family wrapper**.

crc (Cyclic Redundancy Check)
: Used to verify that data has been stored properly; it's used like a checksum, but is less prone to multiple-bit errors. During a transfer, both sides calculate the crc and compare the values, unless the --nocrc option is specified. Enstore uses an Adler 32 crc.

cwd
: current working directory

dCache
: DCache is a data file caching system which acts as an intelligent manager between the user and the data storage facilities. It optimizes the location of staged copies according to an access profile. It decouples the (potentially slow) network transfer rate from the (fast) storage media I/O rate in order to keep the mass storage system from bogging down.

dCap

DCap is a dCache-native C-API access protocol.

DESY

Deutches Elektronen-SYnchrotron; a laboratory in Hamburg, Germany that conducts particle physics research.

direct I/O

Direct I/O differs from normal disk read/writes in that it by-passes the file system's buffer cache. This is achieved by skipping the (normally done) copy that goes from the application memory space to the kernel buffer's memory space. Direct I/O is an SGI/Linux extension. Compare to **memory-mapped I/O** and **POSIX I/O**.

door (for dCache)

A door is a protocol converter (e.g., for FTP, dCap) between clients and internal dCache systems. Each door is associated with a particular port on the dCache server, and has its own access profile.

drivestat server

The drivestat server maintains statistical information of the drives.

ecrc

Stands for Enstore crc (cyclic redundancy check); see **crc**. The ecrc program calculates the crc of a local file.

encp

Encp as an end-user command is considered to be deprecated. It was designed to be used with Enstore, and used to copy data files from disk to storage media and vice-versa. This command is distributed as part of the **encp** product, available from kits under `ftp://ftp.fnal.gov/products/encp/` or `ftp://ftp.fnal.gov/KITS/<OS>/encp/`, e.g., `ftp://ftp.fnal.gov/KITS/Linux/encp/`

Enstore

Enstore is the mass storage system implemented at Fermilab as the primary data store for experiments' data sets. It provides distributed access to data on tape or other storage media both locally and over networks.

enstore.conf

A configuration file to allow for multiple network interface cards dedicated to Enstore, and to map the interfaces to routers. The default location for the file is `/etc/enstore.conf`. The location of the file can be overridden with the environment variable ENSTORE_CONF. See **ENSTORE_CONF**.

ENSTORE_CONF

Environment variable that can be used to override the location of the `enstore.conf` file. See **enstore.conf**.

ENSTORE_CONFIG_HOST

An environment variable which points to the Enstore server that is running the configuration server

ENSTORE_CONFIG_PORT

An environment variable which sets the port number; the value is (by convention) 7500 for all installations at Fermilab.

ensync

A wrapper for encp that allows you to copy the contents of an entire directory structure to Enstore via a single command.

event relay (ER)

The Event Relay is a server that forwards messages based on subscription. All the Enstore servers send messages to the ER. Any server may “subscribe” to the ER in order to have messages of particular types forwarded to it.

fairshare

A mechanism used in Enstore’s queue management that helps to keep any one storage group (experiment or group) from monopolizing tape drives. Fairshare determines which storage groups currently have jobs in progress (at a mover) and which ones don’t, then gives preference to requests associated with those that don’t.

file clerk (FC)

The File Clerk is a server that tracks files in the system. It manages a database of metadata for each data file in the Enstore system.

file family

A file family is a grouping of data; it defines a category, or family, of data files. Each experiment defines a set of file families for its data. A given storage volume may only contain files belonging to one file family.

file family width

File family width is a value used to limit write-accessiblity on data storage volumes. At any given time, Enstore limits the number of volumes associated with a given file family that are open for writing to the value of the file family width.

file family wrapper

A file family wrapper consists of information that gets added to the front and back of data files as they’re written to media, and defines the files’ format on the storage volume. The format of the wrapper depends on the type of wrapper used. (See **cern wrapper** and **cpio_odc wrapper**.)

filemark

A filemark is a physical mark on tape indicating end of file. Tape drives recognize it and can do high speed searches over it.

ftp

File transfer protocol.

gridftp

See **GSI ftp**.

GSI ftp

An implementation of ftp that uses Grid Proxies for authentication and authorization and is compatible with popular tools such as globus-url-copy (from the globus toolkit).

inquisitor

The Inquisitor monitors the Enstore servers, obtains information from them, and creates reports at regular intervals that can be viewed on the web.

job

In Enstore terminology, a job is what a user submits to encp. See **request** for comparison.

Kerberized ftp client

A Kerberized ftp client is an ftp client that implements Kerberos v5 authentication.

layer

Pnfs stores metadata about each file in “layers”, each layer containing a specific type of metadata. Each stored data file has its own set of these layers. Currently, only layers 1 and 4 are used.

library

A library in Enstore is comprised of both the physical data storage media, robotic devices and drives. An Enstore library is typically called a robot.

library manager (LM)

A Library Manager is a server which controls a virtual library. LMs receive requests for file copies from users via **encp** and they distribute the requests to the Movers.

log server (LS)

The Log Server (LS) receives messages from other processes and logs them into formatted log files that are available online.

media changer (MC)

The Media Changer mounts and dismounts the media into and out of drives according to requests from the Movers.

memory-mapped I/O (mmapped I/O)

With this type of I/O, part of the CPU’s address space is interpreted not as accesses to memory, but as accesses to a device; once you map a file to memory, changes made to the memory map are propagated back to the file. Mmapped I/O strives to avoid memory copies of the data between the application memory space and the kernel memory space (also see **direct I/O** and **POSIX I/O**). Mmapped I/O is in the POSIX standard.

monitor server (MS)

The Monitor Server is available for investigating network-related problems. It attempts to mimic the communication between an encp request, the corresponding library manager, and the mover.

mover (MV)

A Mover is a process responsible for efficient data transfer between the **encp** process and a single, assigned media drive in a library (robot). The Mover receives instructions from a Library Manager (LM) on how to satisfy the users’ requests. The Mover sends instructions to the Media Changer (MC) that services the Mover’s assigned drive in order to get the proper volume mounted.

pnfs layer

See “layer”.

pnfs namespace

Pnfs is an independent namespace package, written at DESY. It presents a collection of library database entries as a UNIX-like file system, and thus allows users to browse stored files as though they reside in this file system. Pnfs is mounted like NFS, but it is a virtual file system only. It maintains file grouping and structure information via a set of tags in each directory.

pnfs tags

See tags.

POSIX I/O

POSIX is a name applied to a widely used family of open system standards based on UNIX. POSIX I/O refers to the POSIX standards for I/O.

request

In Enstore terminology, after a user submits a job to encp, encp sends a request to Enstore to process the job. See **job** for comparison.

resubmit

Encp has functionality to retry and resubmit requests, where we distinguish between these two terms. Encp will *resubmit* a request if it has been waiting for a mover for over 15 minutes, but not due to an error condition. See **retry**.

retry

Encp has functionality to retry and resubmit requests, where we distinguish between these two terms. Encp will *retry* (i.e., resend) a request after an error occurs. See **resubmit**.

SRM

SRM (Storage Resource Management) is the middleware for managing storage resources for the grid.

storage group

A storage group is an identifier corresponding to an experiment that Enstore uses as it controls and balances assignment of resources such as tape drives and media. Each storage group (i.e., each experiment) is assigned an area in PNFS.

storage volume

A unit of mass storage, e.g., a tape.

streaming (of files on tape)

Streaming refers to the sequential access of adjacent files on tape at the maximum tape read/write speeds.

striping (of files on tape)

Striping refers to single files (usually large ones) being split onto two or more volumes, each writing simultaneously, in order to expedite the writing process. (Striping is not supported under enstore.)

suspect volume

A volume becomes suspect when a mover communicates to the appropriate library manager that it had a problem with the volume. It is not yet established that the volume is faulty.

tags

Pnfs uses tag files (usually just called tags) in the `/pnfs` namespace to specify file-specific configuration information, and **encp** transfers this information to Enstore. Tags are associated with directories in the `/pnfs` namespace, not with any specific file, and thus apply to all files within a given directory.

virtual library

A Virtual Library (VL) is a subset of an Enstore data storage library. It can contain one and only one type of media. It is paired with its own library manager which controls it.

volume

See storage volume.

volume assert

A job in which a volume (tape) gets mounted and certain attributes are read in order to check, and thus assert, that the volume is "ok" (without actually checking the entire contents). Requesting volume asserts is an administrative task, and these requests are assigned the lowest priority.

volume clerk

The Volume Clerk (VC) is a server that stores and administers storage volume (tape) information.

volume family

The triplet "storage group + file family + file family wrapper" is called a volume family. In order for different data files to be stored on the same volume, all three of these pnfs tags for the files must match.

wrapper

See file family wrapper.

Index